Norges miljø- og biovitenskapelige universitet
Institutt for matematiske realfag og teknologi

Masteroppgave 2015
30 stp

Utvikling av ett dataprogram for design av S-N kurver til utmattingsberegninger

Development of a Computer Program for Design of S-N Curves Used in Fatigue Calculations

Toril Fjeldaas Rygg

Forord

Dette prosjektet er gjennomført som en del av masterprogrammet i Maskin, prosess og produktutvikling ved Institutt for matematiske realfag og teknologi ved NMBU. Prosjektet er gjennomført i vårsemesteret 2015 med et omfang på 30 studiepoeng.

Ved utmattingsberegninger benyttes vanligvis S-N kurver for å beregne utmattingslevetiden. I noen tilfeller må man utføre testing og lage kurvene selv. Årsaken til det kan være at man ikke har en kurve for den legeringen, geometrien eller miljøet som materialet skal operere i.

IMT ved NMBU ønsker å styrke kompetansen innenfor utmattingsberegninger ved blant annet å utvikle sitt eget dataprogram for å fastlegge S-N kurven og har derfor initiert denne oppgaven.

Oppgaven tar for seg bakgrunnen og formelverket for design av S-N kurver, med en gjennomgang av noen relevante standarder. I tillegg er det utviklet et dataprogram som bruker dette formelverket og en brukerveiledning for å forklare hvordan programmet skal brukes.

Rapporten starter med å gi en oversikt over utmatting som tema. Videre følger en gjennomgang av relevante standarder på området hvor formelverket med bakgrunn er satt opp i ett eget statistikkkapittel. Det er her også hentet inn ekstern statistikklitteratur. Oppgaven avsluttes med en gjennomgang av utviklingen av dataprogrammet.

Oppgaven kan brukes som en veiledning for få en innføring i utmatting og statistikken som er nødvendig for å forstå bakgrunnen for S-N kurvene. Det utviklede dataprogrammet vil sammen med brukerveiledningen gjøre leseren i stand til å benytte data fra testing og lage egne S-N kurver.

Det forutsettes i oppgaven en generell kjennskap til utmatting og statistikk på ingeniørnivå. For fullt utbytte av kapittel 5.5.3 kreves det en mer inngående kjennskap til statistikk og matriseregning, og kapittel 6.3 krever god kjennskap til programmering.

Jeg vil rette en spesiell takk til førsteamanuensis Trygve Almøy for god støtte til statistikkdelen i oppgaven. Jeg vil også takke veileder førsteamanuensis Geir Terjesen for den interessante oppgaven og god oppfølging underveis.

Ås, den 12. mai 2015

Toril Fjeldaas Rygg

Sammendrag

Man må til tider lage egne S-N kurver for å beregne utmattingslevetid, noe som ofte innebærer å sette opp formlene fra relevante standarder manuelt i ett regneark. Det ønskes en gjennomgang av disse utmattingsstandardene slik at disse formlene kan identifiseres. Videre ønsker IMT ved NMBU ett enkelt dataprogram som kan bearbeide utmattingsdata og gi ut ønskede resultater som S-N kurver og feilestimater.

Målet med rapporten er å gi en oversikt over noen relevante standarder og formelverk som kan benyttes når S-N kurver skal designes. Videre er målet å utvikle et dataprogram med brukerveiledning som kan beregne S-N kurver med noen tilhørende feilestimater.

Prosjektet har blitt utført som en litteraturstudie om beregning av S-N kurver og feilestimater. Videre er det utført dataanalyser og praktisk testing i forbindelse med utvikling av dataprogrammet. Programmet har også blitt brukertestet i liten skala.

Generelt kan man si at beregning og vurdering av S-N kurven i stor grad kan hentes direkte fra lineær regresjonsanalyse, mens krav til designlinjen varierer mellom standardene.

ASTM [5] kommer med anbefalinger til antall tester og stiller krav til testdataen, og tar ellers for seg den statistiske bakgrunnen for å beregne S-N kurven. ASTM går ikke inn på krav til designlinjen.

DNV [6] viser til ekstern litteratur for beregning av selve S-N linjen, men stiller med to mulige designlinjer. Disse legger vekt på om man kan anta at standardavviket er kjent for datasettet, hvor linjen for ukjent standardavvik er betydelig mer konservativ. Designlinjen er også avhengig av antall prøver.

Britisk Standard [7] går heller ikke inn på beregning av S-N linjen i større grad og har en enkel tilnærming til designlinjen hvor de kun ser på sannsynlighet for brudd ut i fra antall standardavvik man trekker fra middellinjen. De setter 2 standardavvik som normalen for designlinjen og korrigerer ikke for usikkerhet fra få prøver.

International Standard [8] oppgir ett sett med formler basert på lineær regresjonsanalyse og stiller med en formel for designlinjen basert på en ikke-sentral Student t-fordeling. Denne er i utgangspunktet ikke lineær, men kan forenkles til å følge vanlig form.

I tillegg til de fire standardene har to rapporter fra Sintef [9, 10], og en fra IIW [11], blitt vurdert. Fra disse er det brukt en formel for designlinjen som likner den fra ISO, men hvor det brukes en vanlig Student t-fordeling.

Fra litteraturen er det også tatt med statistiske vurderinger av S-N linjen, samt statistisk vurdering av to ulike datasett mot hverandre.

Resultatene fra dataprogrammet er sammenliknet med kjente resultater for å sikre at formlene er implementert på en riktig måte.

Programmet fungerer på en tilfredsstillende måte og sees på som en betydelig forbedring til dagens metode med bruk av regneark. Tilbakemeldinger fra brukere som har prøvd programmet indikerer at programmet er enkelt å bruke. Programmet tilbyr enkel analyse av dataen med god grafisk kvalitet på plottene og med resultater som er verifisert som riktige.

Brukeren står fritt til å velge standard for designlinjen i programmet, men ut i fra bakgrunnen for beregning sees DNV og IIW på som gode og trygge valg.

Summary

One must occasionally make S-N curves to calculate fatigue life oneself, which in many cases means writing the formulas from relevant standards manually in a spreadsheet. IMT at NMBU want a review of these standards so that these formulas can be identified, as well as a simple computer program that does calculations on the fatigue data and give out results like S-N curves and error estimates.

The goal of this project is to give an overview of some of the relevant standards and of the formulas that can be used in the design of S-N curves. Further, the goal is to develop a computer program with a user manual that can calculate S-N curves with error estimates.

The project have been conducted as a literature study on the background for calculating the S-N curves with error estimates. It has also been carried out data analyses and practical tests during the development of the program. The program has also been user tested in a small scale.

Generally, calculation and evaluation of the S-N curve can be found directly from linear regression analysis, while the demands for the design line varies between the standards.

ASTM [5] gives recommendations for the number of tests needed and has a list of demands for the test data, as well as a detailed overview of the statistical background for calculating the S-N curve. ASTM does not give recommendations for the design line.

DNV [6] refers to external literature for the calculation of the S-N line itself, but has two possible options for the design line. These depend on whether or not the variance of the model can be assumed known, where the line for an unknown variance is significantly more conservative. The design line is also dependent on the number of tests performed.

British Standard [7] does not say much about the calculations of the S-N line either and has a simple approach to the design line, where they only looks at the probability of fracture based on the number of standard deviations away from the mean line. They give 2 standard deviations as the norm for the design line and does not correct for insecurities from few samples.

International Standard [8] gives a set of formulas based on linear regression and gives a formula for the design line based on a non-central Student t-distribution. This does not give a linear line, but can be simplified to follow the regular form.

Two reports from Sintef [9, 10], and one report from IIW [11], has been considered in addition to the four standards. These give a formula for the design line similar to ISO, but with a regular Student t-distribution, that has been included as well.

Statistical evaluations of the S-N line and statistical evaluation of two datasets compared to each other has also been included from the literature.

The results from the program is compared with known results to make sure the formulas are implemented in a correct way.

The program works in a satisfying way and is considered a significant improvement to the use of spreadsheets. The feedback from users that have tested the program indicates that the program is easy to use. The program offers a simple analysis of the data with a good graphical quality to the plots and with results that has been verified as correct.

The user can choose the design line in the program freely, but DNV and IIW seems like good and safe choices from looking at the background for the calculations.

Innholdsfortegnelse

Side:

1. Innledning

1.1. Bakgrunn

Ved utmattingsberegninger benyttes vanligvis S-N kurver for å beregne utmattingslevetiden.

I mange tilfeller må man lage S-N kurven selv fordi kurven ikke eksisterer for det tilfellet man jobber med, enten det er fordi legeringen, geometrien eller miljøet legeringen skal operere i avviker fra den S-N kurven man har tilgjengelig. S-N kurver lages fra testdata, hvor man måler levetid ved ulike spenningsvidder. Dataene behandles så statistisk slik at man får en kurve med riktig sikkerhet til bruk.

I dag lager man ofte kurvene i et regneark, hvor formlene legges inn separat for den standarden man benytter og kurvene tegnes med varierende kvalitet. Det ønskes ett enkelt program som kan bearbeide denne dataen og gi ut ønskede resultater som S-N kurver og feilestimater.

1.2. Problemstilling

Utvikling av ett dataprogram for design av S-N kurver til utmattingsberegninger.

1.3. Målsettinger og begrensninger

Hovedmål:
Lage et dataprogram for design av S-N kurver etter ønsket standard, samt lage en oversikt over gjeldende regelverk og det statistiske teoretiske grunnlaget for programmet.

Delmål:
- Lage en oversikt over gjeldende regelverk for konstruksjon av S-N kurver.
- Lage en oversikt over statistisk teoretisk grunnlag.
- Lage et dataprogram for bearbeiding av måledata med spredningdiagram, regresjonslinje og designkurve i et dobbel-logaritmisk diagram med tilbakemelding om statistisk gyldighet.
- Lage en brukermanual til dataprogrammet.
- Kontrollere dataprogrammet mot kjente resultater.
- Lage en rapport som tar for seg resultatene.

Begrensninger:
- Programmet vil kun lages for Windows-brukere.
- Programmet vil kun ta for seg lineære S-N kurver uten knekk innenfor 10^4 og $5{\cdot}10^6$ sykler.

1.4. Tids og arbeidsplan med milepæler

Det ble satt opp en arbeidsplan med milepæler basert på delmålene i kapittel 1.3 (se Vedlegg 1). Da det er vanskelig å estimere tidsbruk for programmering med dette omfanget ble det satt opp en forholdsvis tidlig dato for ferdigstilling, men med mulighet for å bruke mer tid på dette området eller utvide omfanget av programmet.

1.5. Metodebruk og løsningverktøy

1.5.1. Utviklingsverktøy

Pugh [1, 2]:
Pughs metode er et verktøy for å sammenligne ulike løsninger for å finne den som best møter ett sett med kriterier. Disse løsningene vurderes ofte mot et referanseprodukt og vurderes ut i fra om de stiller sterkere, svakere eller likt som referansen (+ / - / =). Hvis man ikke har ett referansepunkt vurderes løsningene i større grad relativt til hverandre og hvorvidt de oppfyller kriteriene bedre eller dårligere.

En styrke med Pughs metode er at den kan ta inn mange ulike kriterier og gi en totalt poengsum ut i fra disse. På denne måten kan man velge det beste konseptet ut i fra hva slags score den har fått, noe som gir en mer nøytral og pålitelig utvelgelse.

1.5.2. Programvare

Python 2.7 er ett fleksibelt og kraftig programmeringsspråk med åpen kildekode. Det kan brukes og distribueres fritt, også kommersielt, og har gratis programlisens.

wxPython 3.0 er en innpakning av wxWidgets (som er skrevet i C++) og er ett sett med verktøy for programmering av grafisk brukergrensesnitt. wxPython fungerer som en utvidelsemodul for Python og har også gratis programlisens.

py2exe 2.5 er en utvidelse til Python som gjør en Python fil (.py) om til en kjørbar .exe fil som kan kjøre på Windowsmaskiner uten Python installert.

MikTeX Portable 2.9 er en distribusjon av LaTeX, et system for tekstoppsett. MikTeX består av verktøy for å forberede og bearbeide dokumenter ved bruk av oppsettspråket TeX/LaTeX og de tilbyr en bærbar versjon som ikke krever installering. MikTeX kan også fritt distribueres.

Adobe Creative Suite CS6 inkluderer InDesign og Illustrator, som er programmer for oppsett og design av grafiske elementer.

1.6. Kvalitetssikring

Rapporten:
Rapporten skal settes opp i InDesign, hvor mest mulig av referanser og nummereringer gjøres automatisk. Videre skal det lages stiler for ulike typer innhold for å få mest mulig kontroll med oppsettet. Til slutt vil det gjøres en manuell gjennomgang av rapporten to ganger for å kontrollere at alle refe-

ranser, nummereringer og stiler stemmer. Alle formler skal kontrolleres to ganger og alle symboler skal kontrolleres mot symbolliste.

Produkt:

- Kravspesifikasjonene skal diskuteres/tolkes i samråd med oppdragsgiver på jevnlige møter.
- Når produktet nærmer seg ferdigstilling skal det brukertestes mot en liten gruppe for å få tilbakemelding om brukervennlighet og eventuelle feil og mangler. Programmet skal også testes med mål å fremprovosere feil ved å gi ugyldig input med mer.
- Videre skal resultatene fra programmet sammenlignes mot kjente resultater fra både håndberegninger og andre programmer.

1.7. Symboler

Tabell 1: Symboler og enheter brukt i oppgaven

Symbol	Betydning	Enhet
σ	Spenning.	MPa
$\Delta\sigma$	Spenningsvidde[1].	MPa
σ_a	Spenningsamplitude.	MPa
σ_m	Middelspenning.	MPa
$\Delta\sigma_i$ / σ_i	Spenningsvidde for prøve nr. i.	MPa
F	Kraft.	N
N	Antall spenningssykler.	-
N_{Design}	Designkurve.	-
$N_{1-p/1-P,1-p}$	Designkurve med 1-p eller 1-P, 1-p sikkerhet.	-
N_i	Antall spenningssykler for prøve nr. i.	-
n	Antall prøver.	-
l	Antall spenningsnivåer.	-
i	Indeks for nummerering.	-
k_i	Antall prøver på spenningsnivå nr. i.	-
j	Antall estimerte parametere.	-
g	Indeks for datasettnummerering.	-
r	Replikasjon for testen.	%
α	Skjæringspunktet for S-N kurven.	-
β	Stigningen til S-N kurven.	-
$\hat{\alpha}$	Estimat for α.	-
$\hat{\beta}$	Estimat for β.	-

1 *σ brukes til tider i stede for $\Delta\sigma$ der det ikke er grunnlag for misforståelse.*

Tabell 1 forts:

Symbol	Betydning	Enhet
C	10^{α}.	-
$\overline{C}$	C korrigert for designlinjen.	-
m	-β.	-
X	Logaritmen til Δσ, enten enkeltverdi eller matrise.	-
Y	Logaritmen til N, enten enkeltverdi eller matrise.	-
X_i	X-verdi nr. i blant n prøver.	-
Y_i	Y-verdi nr. i blant n prøver.	-
e_i	Residual nr. i.	-
d_i	Standardisert residual nr. i.	-
$\overline{X}$	Gjennomsnittlig X-verdi blant prøvene.	-
$\overline{Y}$	Gjennomsnittlig Y-verdi blant prøvene.	-
$\hat{X}$	Estimert X-verdi, enten enkeltverdi eller matrise.	-
$\hat{Y}$	Estimert Y-verdi, enten enkeltverdi eller matrise.	-
S	Standardavvik.	-
$\hat{S}$	Estimert standardavvik.	-
SD	Alternativ skrivemåte for estimert standardavvik[2].	-
S^2	Varians.	-
$\hat{S}^2$	Estimert varians.	-
b	Antall standardavvik designlinjen korrigeres med.	-
θ	Matrise med parametere.	-
$\hat{\theta}$	Matrise med estimerte parametere.	-
H	Hattematrisen.	-
I	Identitetmatrisen.	-
H_0	Nullhypotese.	-
H_1	Alternativ hypotese.	-
p	Akseptabel usikkerhet i et (1-p) konfidensintervall eller ved ett (1-p) konfidensnivå.	% [1]
P	Prosentverdi i 1-P pålitlighet til estimat.	% [1]
P_i	Kumulativt sannsynlighetspunkt nr. i.	% [1]
RSS	Summen av kvadrerte residualer.	-
v	Antall frihetsgrader.	-
$t_{p, v}$	t-verdi fra Student t-fordeling med p usikkerhet og v frihetsgrader.	-

1 *Prosentverdier brukes vekslende på prosent og på desimalform.*
2 *SD fremstår mer lettlest/tydelig for brukeren og brukes i presentasjon/valg av resultater.*

Tabell 1 forts:

Symbol	Betydning	Enhet
$f_{p, v1, v2}$	f-verdi fra fisherfordeling med p usikkerhet og v1 og v2 frihetsgrader.	-
$k_{1-P,1-p, v}$	Koeffisient for ensidig toleransegrense for normal-fordeling med 1-P % pålitelighet til estimeringen og 1-p konfidensnivå til estimatet.	-
r^2	Korrelasjonskoeffisient. Beskrivelse av kvaliteten til forklaringsmodellen.	-

1.8. Definisjoner/Terminologi

Nedenfor følger en oversikt over uttrykk brukt i oppgaven.

Tabell 2: Uttrykk med definisjoner

Uttrykk	Definisjon
ASTM	ASTM International, tidligere American Society for Testing and Materials.
Brukergrensesnitt	Kontaktflaten mellom bruker og maskin som gjør at brukeren kan styre programmet.
BS	British Standard
Designkurve / linje	S-N kurve korrigert for designformål for å gi en bestemt sikkerhet mot brudd.
Designliv	Levetid der det er liten sannsynlighet for at brudd vil opptre.
DNV	Den Norske Veritas, nå DNV GL.
Forklaringsvariabel	Variabelen som blir kontrollert i forsøket.
Frihetsgrader	Antall observasjoner i testen minus antall estimerte parametre.
Hypotesetest	Statistisk metode for å teste hypoteser/teorier om en eller flere populasjoner med indre variasjon. Man setter opp en nullhypotese, H_0, og en alternativ hypotese, H_1, og tester om datamaterialet gir grunnlag for å forkaste nullhypotesen med en ønsket grad av sikkerhet.
Konfidensnivå	Grense der man med 1 - p sannsynlighet kan forvente at verdiene inntreffer.
Konformitet	Samsvar i form.
Kumulativ	Verdier som samles opp/eskalerer. Verdiene bygger på tidligere verdier til en total sum.
ISO	International Standard Organization
Logaritme (log)	Logaritmen til et tall er det tallet man må opphøye grunntallet i (10 hvis ikke noe annet er spesifisert) for å få tallet.
Populasjon	Totalt antall prøver av en gitt type.
Prøve	En eller flere elementer fra en populasjon for testing.

Tabell 2 forts:

Uttrykk	Definisjon
Regresjonskurve / linje	Den linjen som best forklarer datasettet. Linjen tilsvarer 50 % sannsynlighet for brudd.
Replikasjon	Repetisjon av et sett med verdier for sammenligning i en test.
Residual	Avvik fra forventet verdi.
Responsvariabel	Variabel avhengig av forklaringsvariabelen.
Runout	En prøve som ikke har gått til brudd i løpet av testen.
S-N kurve	Grafisk representasjon av sammenhengen mellom utmattingslevetid N og spenningsvidde $\Delta\sigma$.
Snitt / Gjennomsnitt	Sum av alle verdier delt på antall observasjoner.
Spenningsvidde $\Delta\sigma$	Differansen mellom største og minste spenning, dvs. to spenningsamplituder.
Standardavvik	Mål for spredning i datasettet. Roten av variansen.
Utmatting	Gradvis svekking av et materiale fra initiering og vekst av sprekker forårsaket av syklisk belastning.
Utmattingsbrudd	Brudd forårsaket av gradvis svekkelse av tverrsnittet i et materiale fra utmatting.
Utmattingsgrense	Spenningsvidde der man antar uendelig levetid for materialet.
Utmattingslevetid N	Antall sykler en prøvestav utsettes for før brudd inntreffer.
Utmattingsstyrke	Spenningsvidden hvor materialet vil gå til brudd ved en gitt utmattingslevetid.
Variabel	Vilkårlig element i en mengde betegnet med en bokstav.
Varians	Mål for variasjon i datasettet.

2. Produktspesifisering

2.1. Kravspesifikasjoner

Kravspesifikasjonene er delt mellom rapportkrav, funksjonskrav og designkrav. De to førstnevnte vil fungere som retningslinjer for utvikling av oppgaven, mens designkravene vil brukes til direkte utvelgelse av løsning for programmet.

2.1.1. Rapport

Rapporten skal dekke følgende områder:

- **Gjennomgang av standarder:** Rapporten skal inneholde en gjennomgang av relevante standarder innen bearbeiding av måledata fra S-N testing og krav til designkurver.
- **Gjennomgang av statistisk grunnlag:** Den statistiske metoden for bearbeiding av måledata skal gjennomgås og eventuelle ulikheter mellom standarder, rapporter og fagbøker skal vurderes.
- **Verifisering:** Resultatene fra programmet skal verifiseres mot håndberegninger og kjente resultater fra andre programmer eller annen litteratur.
- **Brukermanual:** Rapporten skal også inneholde en veiledning til dataprogrammet.

2.1.2. Funksjonalitet

Programmet skal ha følgende funksjonalitet:

- **Dataområde:** Programmet skal konstrueres for å bearbeide måledata i området mellom 10^4 og $5{\cdot}10^6$ sykler.
- **Datainput:** Brukeren skal kunne legge inn data manuelt, men også hente data fra tidligere kjøring.
- **Lagring:** Brukeren skal kunne lagre og hente inn data/innstillinger.
- **Valg av standarder:** Brukeren skal kunne velge standard for beregning av designlinje.
- **Plot:** Programmet skal kunne vise og lagre plot med god kvalitet.
- **Valg av output:** Brukeren skal kunne velge hva som skal vises av resultater/plot.
- **Statistisk evaluering:** Programmet skal gi statistiske evalueringer av resultatene med konfidensintervall, gyldighet av modellen og annen relevant informasjon.
- **Rapport:** Brukeren skal kunne eksportere en rapport/tekstfil med resultater.
- **Kjøring:** Programmet skal kunne enkelt kjøres på PC-er av vanlige brukere.

- **Bruk:** Programmet bør kunne brukes med kun et par tastetrykk, men også ha mulighet for mer spesialiserte innstillinger.
- **Tilbakemelding om feil/feilbruk:** Programmet bør gi brukeren relevant informasjon om feil som oppstår (f.eks. hvis brukeren oppgir ugyldige verdier for dataen) eller hvis brukeren prøver å bruke programmet feil (f.eks. har testdata utenfor dataområdet programmet støtter).

2.1.3. Design

Designet til programmet (brukergrensesnittet) bør oppfylle følgende krav for brukervennlighet og fleksibilitet:

- **Programflyt:** Bruken av programmet bør følge en naturlig flyt og det bør være tydelig hva som skal/bør skje først og sist.
- **Gjenkjennbart:** Programmet bør bruke en programstruktur/programelementer som brukeren vil kjenne igjen fra andre programmer.
- **Oversiktlighet:** Dataen bør presenteres på en ryddig og oversiktlig form, uten for mye informasjon på en gang. Samtidig må ikke viktig informasjon gå tapt/oversees.
- **Enkelhet:** Brukeren bør ikke trenge å forholde seg til spesialinnstillinger hvis det ikke er ønskelig.
- **Utvidbart:** Programmet bør ha et design som gjør at det kan utvides med mer funksjonalitet senere.

Ved utvikling av brukergrensesnittet bør designet skje under forventning om at brukeren ikke vil ta seg bryet med å lese brukermanualen og designet bør derfor lages mest mulig selvforklarende.

2.2. Konseptdrøfting

Funksjonen til programmet kan ved første øyekast deles inn i tre hovedområder: legge inn data, vurdere resultater og eksportere resultater. Av disse tre områdene er det kun den første som er et krav for bruk, siden brukeren kan stoppe etter vurdering av resultatene, eller forvente gode resultater og be om eksport med en gang hvis programmet tillater det. Videre vil resultatene kunne deles i to elementer: grafisk fremstilling av linjer og tekstbaserte resultater med verdier for linjen og statistikk. Tar man inn muligheten for brukeren å endre innstillinger underveis får man ett mye mer komplekst bilde.

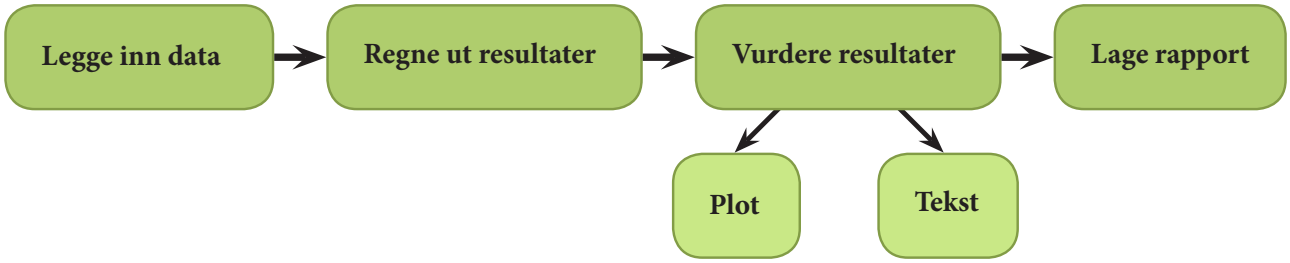
Et dataprogram er egentlig bare ett sett med kommandoer og kan kjøres direkte med et tekstbasert brukergrensesnitt. En svakhet med denne løsningen er at den enten må settes opp med en veldig lineær programstruktur (dvs. at ting må skje i en spesifikk rekkefølge), noe som vil virke veldig hemmende, eller så må brukeren gi inn ett sett med gyldige kommandoer, noe som er bruksmessig krevende.

Et grafisk brukergrensesnitt er mer gjenkjennbart for brukeren, og har som regel en lavere terskel for bruk. En grafisk løsning kan også gjøres veldig fleksibel, uten at det blir forvirrende for brukeren. Et grafisk brukergrensesnitt er derfor absolutt å foretrekke.

Programmeringsspråket for oppgaven må egne seg til databehandling, men også støtte grafisk brukergrensesnitt. Det er mange programmeringsspråk å velge mellom, men da undertegnede har god kjennskap til programmering i Python er Python et naturlig valg, sammen med utvidelsen wxPython for det grafiske.

2.3. Formgivning og design

Programmet må designes ut i fra bruken, som grovt sett kan settes opp på denne måten:

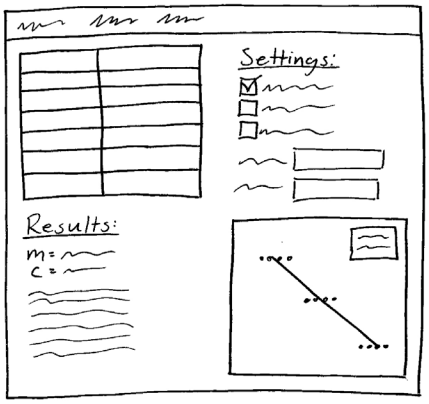


Figur 1: Arbeidsflyt for bearbeiding av måledata.

Når det gjelder boks 2 og 4, "Regne ut resultater" og "Lage rapport", er dette funksjoner som programmet skal utføre og vil ikke nødvendigvis gi mye tilbakemeldinger til brukeren i brukergrensesnittet. Det vil derimot være nødvendig at brukeren kan gi innstillinger til disse funksjonene.

Videre kan det vurderes om resultatene skal fremstilles samlet eller hver for seg.

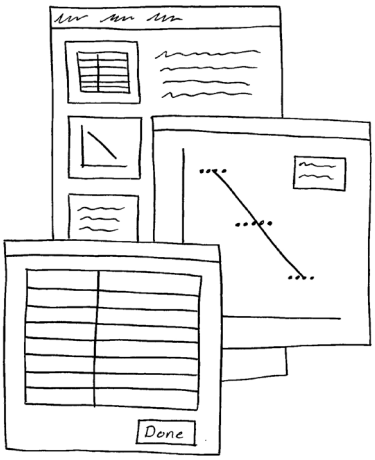
Et grafisk brukergrensesnitt som legger til rette for overnevnt bruk kan implementeres på flere ulike måter. Tre løsninger vurderes nedenfor:



Figur 2: Samlet løsning.

Samlet (Alt i ett vindu)
Her vil vinduet deles inn i ulike felt som inneholder ulik innput eller informasjon, med en naturlig arbeidsflyt fra venstre mot høyre i hver rad nedover.

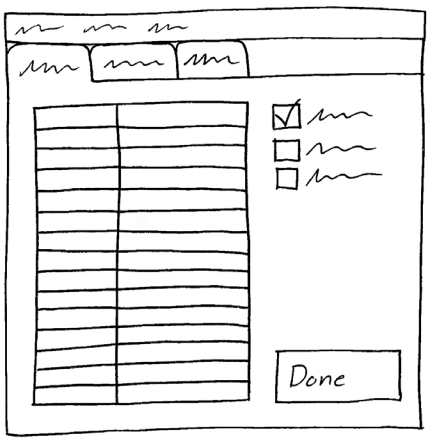
En svakhet med denne løsningen er begrenset plass og dermed begrenset mulighet for å legge til ekstra funksjonalitet, samt at designet kan fremstå rotete.



Figur 3: Flere vinduer.

Flere vinduer
Med denne løsningen vil man ha ett hovedvindu med oversikt over de ulike funksjonene til programmet, slik som å legge inn data og se plot, og knapper som åpner disse funksjonene i ett eget vindu. På denne måten får man bedre plass og trenger ikke å forholde seg til alt med en gang.

En ulempe er at det kan virke forstyrrende med mange ulike vinduer og brukeren kan føle at designet er uoversiktlig.



Figur 4: Faneløsning.

Faner

Her videreføres tanken med å skille de ulike funksjonene, men samtidig beholder man en mer strukturert form. Her vil det være en naturlig flyt å starte med første fanen og forflytte seg bortover, med muligheten til å enkelt gå tilbake når som helst for å gjøre endringer.

Denne løsningen forutsetter at brukeren er komfortabel med en fanebasert løsning, slik man ser i for eksempel nettlesere.

Valg av løsning

Designløsningene er satt opp i en Pugh matrise (se 1.5.1) for å vurdere hvilke løsning som er best egnet. I vurderingen er kriteriene fastsatt under 2.1.3 benyttet.

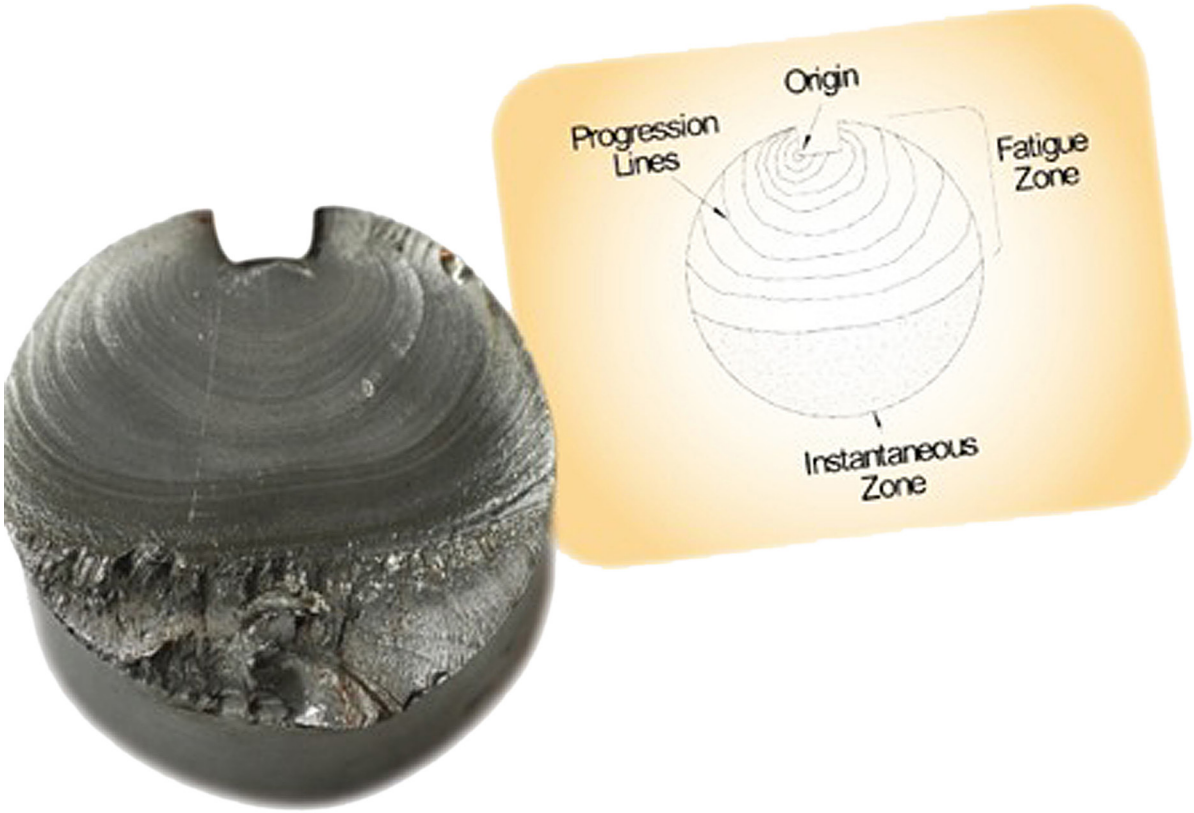
Tabell 3: Vurdering av løsningene med en Pugh matrise.

	Ulike design		
Kriterier	**Samlet**	**Flere vinduer**	**Faner**
Programflyt	+	-	+
Gjenkjennbart	+	+	=
Oversiktlig	-	-	+
Enkelt	+	-	+
Utvidbart	-	+	+
TOTALT	+1	-1	+4

Fra denne vurderingen ser man at løsningen med faner kommer best ut. Denne løsningen vil derfor benyttes videre i utviklingen av programmet. Flere vinduer kommer dårligst ut, men det kan være aktuelt å bruke metodikken med ekstra vinduer for ekstra funksjonalitet som innstillinger og informasjonsvinduer.

3. Generelt om utmatting

Utmatting i materialer kommer fra sprekkdannelse og sprekkvekst under dynamiske belastninger og oppstår gjerne i områder med lokalt høye spenningskonsentrasjoner. Sprekkene initieres og vokser ved spenninger langt under materialets strekkfasthet og reduserer tverrsnittet som tar opp kreftene. Dette kan over tid føre til brudd når antall belastningsvekslinger blir stort nok.



Figur 5: Ett typisk utmattingsbrudd med initiering fra spenningskonsentrasjon, utmattingssone med linjer fra belastningssyklus og ett restbrudd [15].

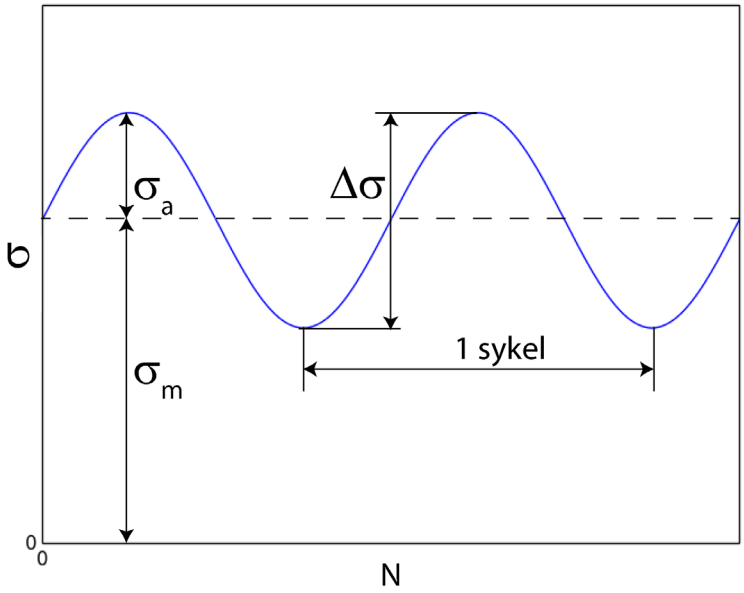
Utmattingslevetid beregnes i stor grad ut i fra data fra utmattingstesting av ulike materialer og geometriske former som belastes på ulike måter. Disse resultatene plottes i et dobbelt-logaritmisk diagram, hvor man plotter spenningsvidde mot antall sykler til brudd.

Noen jernholdige materialer har en nedre spenningsgrense for uendelig levetid, en grense som ofte oppstår mellom 10^6 og 10^7 sykler. Man gjør gjerne den konservative antagelsen at et materiale ikke må belastes over denne utholdenhetsgrensen hvis materialet skal vare mer enn 10^6 sykler [3].

Siden utmatting kommer fra lokale punkt med relativ svakhet, vil resultatene ha en betydelig større

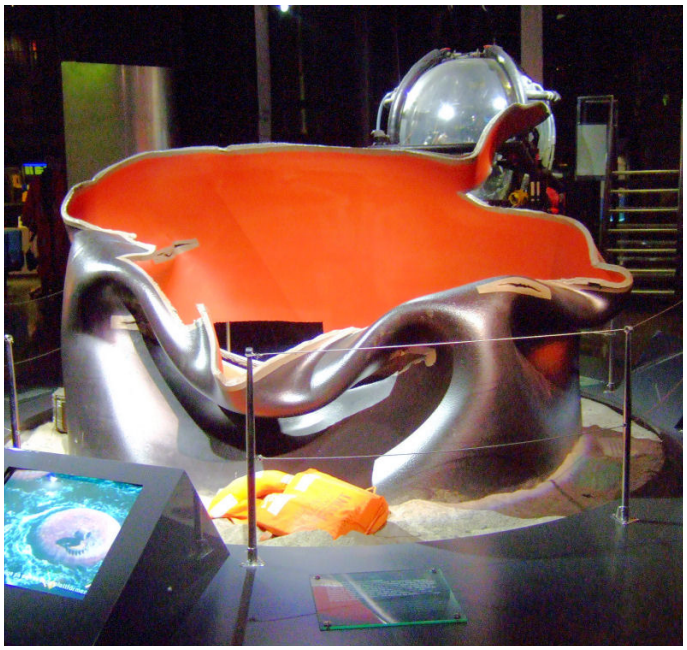
spredning enn resultatene fra en statisk test [3]. Det vil også kunne være store variasjoner i materialkvalitet ut i fra produksjonssted. Videre er det kostbart og tidskrevende å teste mange prøver, så antall tester er ofte begrenset. Den statistiske metoden for å beregne styrken blir derfor av større viktighet.

Utmattingslevetiden er hovedsakelig avhengig av spenningsvidden den utsettes for i form av en syklisk belastning (Δσ), og antall spenningssykler (N). Utmatting er i noen grad også avhengig av middelspenningen, men middelspenningen tas ikke med i vanlige S-N kurver, og er heller noe man korrigerer for senere.



Figur 6: Illustrasjon av dynamisk belastning, hvor σ_A er spenningsamplitude, σ_M er middelspenning og $\Delta\sigma$ er spenningsvidde.

3.1. Viktigheten av utmattingstesting



Figur 7: Del av staget som brakk på A. Kielland platformen, utstilt på Norsk Oljemuseum [16].

Frem til midten av 1800-tallet ble vekslende belastninger behandlet på samme måte som statiske belastninger, med unntak av at man brukte en større sikkerhetsfaktor. En alvorlig togulykke i Verseillle i 1942, hvor det anslås at mer enn 100 døde etter at en av togets akslinger røk, startet en mer systematisk undersøkelse av bakgrunnen for slike brudd. Siden den gang har kunnskapen om utmattingsbrudd økt betraktelig, men ikke uten betydelige kostnader i form av flere store ulykker. Av de mer alvorlige kan man nevne Alexander Kielland ulykken, en boligplatform som i 1980 veltet i Nordsjøen på grunn av utmattingsbrudd i ett stag hvor det var ettermontert en hydrofon. Ulykken krevde 123 liv.

Undersøkelser i Europa og USA viser at mellom 80 og 95 % av brudd i maskinkomponenter under normal drift kommer fra utmatting [4]. Disse bruddene oppstår ofte uten forvarsel og kan skape farlige situasjoner. Samtidig kan man i mange tilfeller ikke legge inn for store sikkerhetsmarginer, da det vil gi for dyre og tunge konstruksjoner. Spesielt i flyindustrien er dette en utfordring. God kunnskap om utmatting og sikker levetid er derfor viktig.

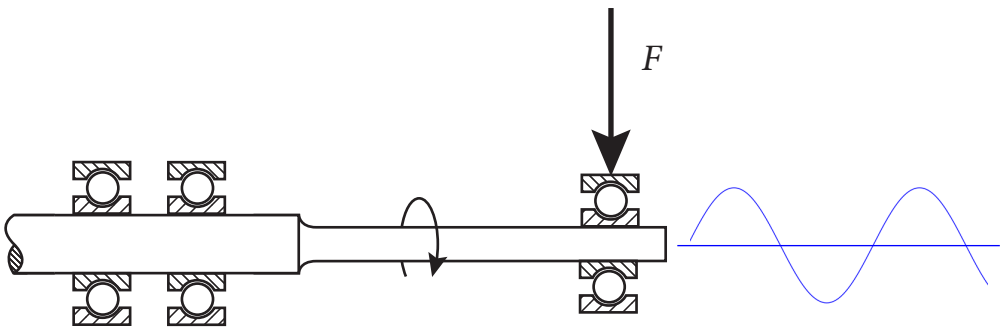
3.2. Dimensjonering mot utmatting

Det finnes flere måter å dimensjonere en struktur mot utmatting:

- **Infinite Life Design:** Her designer man hele strukturen for uendelig levetid, dvs. at ingen belastninger skal overstige utmattingsgrensen der utmatting kan være ett problem. Med denne metoden er det lite sannsynlig at brudd vil oppstå, men som nevnt tidligere kan det være en utfordring for f.eks. flyindustrien hvor de prøver å minimere materialbruk for å senke vekt og kostnader.
- **Safe Life:** Med denne metoden designer man konstruksjonen for en spesifikk levetid, det vil si at man beregner en trygg levetid hvor det er lite sannsynlig at det oppstår brudd. Her bruker man ofte designlinjer som gir en 97,7 % sikkerhet mot brudd, mens sikkerheten i realiteten kan være høyere da man kombinerer dette med lastfaktorer.
- **Fail Safe:** Her designer man konstruksjonen slik at den vil tåle ett utmattingsbrudd. Dette gjelder for statisk ubestemte strukturer hvor kreftene vil omfordeles til andre områder. Med regelmessig kontroll kan man reparere brudd og fortsatt bruke konstruksjonen.
- **Damage Tolerant Design:** Med denne metoden designer for at strukturen skal tåle skade og disse skadene la seg oppdage ved inspeksjon før det blir brudd. Det er her viktig med forholdsvis seige materialer slik at skaden blir lett synlig før det er bruddfare og å ha jevne inspeksjoner.

3.3. Utmattingstesting

Utmattingstesting foregår ved at man tester et tilfeldig og representativt utvalg prøver ved ulike spenningsamplituder og ser hvor mange sykler det tar før prøvene går til brudd. Det er vanlig å teste på tre jevnt fordelte spenningsamplituder, eventuelt fire for store områder, og kjøre flere prøver på hvert nivå. En vanlig utmattingstest er med en roterende aksel som trykkes ned med ulik kraft i den ene enden. Dette vil gi trykk på den ene halvdelen av tverrsnittet og strekk på den andre. Ved rotasjon vil man få en kontinuerlig veksling mellom trykk og strekk i materialet.



Figur 8: Illustrasjon av roterende utmattingstesting med fremstilling av belastningen.

Det går også an å teste for rent trykk/strekk uten rotasjon, men det er som regel mer tidskrevende.

Resultatene fra testingen kan bruke for å beregne en SN-kurve for materialet.

3.4. S-N kurven

S-N kurven beregnes med lineær regresjonsanalyse hvor kurven bestemmes som den linjen som minimerer de totale kvadratiske avvikene fra måledataen. Denne kurven representerer 50 % sannsynlighet for at maskindelen skal tåle et gitt antall sykler ved en bestemt spenningsamplitude. Med statistiske modeller kan man så beregne en designlinje med ønsket sikkerhet mot brudd. Mer om statistisk evaluering av testdataen kommer i kapittel 5.

S-N kurven plottes typisk i ett dobbelt-logaritmisk diagram, dvs. et diagram hvor begge aksene bruker logaritmisk skala, og fremstår da som en rett linje med formen $N = c \cdot \Delta\sigma^{-m}$. Det er verdt å merke seg at S-N kurver plottes med aksene motsatt av normen. Dvs. at spenningen ($\Delta\sigma$), som er forklaringsvariabelen, plottes langs den vertikale aksen, mens antall sykler (N), som er responsen, plottes langs den horisontale aksen. Her tilsvarer -m stigningen og log(c) krysningspunktet med N-aksen ($\Delta\sigma = 10^0=1$).

En logaritmisk skala er en ikkelineær skala som brukes når vidden på verdier er stor. Logaritmen til verdiene vil få en lineær form (10^4, 10^5, 10^6 etc.), mens vanlige tallverdier (200, 300, 400 etc.) får en spredning med ulik intervallstørrelser, hvor avstanden kommer fra logaritmeverdien til tallet.

Figur 9: Eksempel på mellominndeling av en logaritmisk skala.

4. Gjeldende regelverk for beregning av S-N kurven

Det finnes flere standarder og en rekke rapporter som tar for seg utmatting. Flere av disse går også inn på hvordan man beregner en S-N kurve ut i fra måledata og man kan finne ulike anbefalinger til sikkerhet for design. I dette kapittelet vil det gås gjennom noen relevante standarder, hvor formler og uttrykk er skrevet om så de følger symbolbruk og terminologi gitt i kapittel 1.5.

4.1. ASTM E739-10 [5]

Standarden tar for seg S-N og ε-N (*tøyning vs. antall sykler*) forhold som kan tilnærmes en rett linje for et spesifikt intervall av belastning og gir retningslinjer for modellering og analyse. I denne oversikten brukes kun retningslinjene for S-N kurver.

4.1.1. Antagelser/Begrensninger[1]

Det er antatt at logaritmen til utmattingslevetiden er normalfordelt og at variansen av log-levetiden er konstant over hele testområdet.

Standarden gjelder ikke for datasett der man har avbrutte tester hvor prøven ikke har gått til brudd.

4.1.2. Anbefalinger

Siden tilnærmingen skjer innenfor et spesifikt intervall av spenningsvidder, og siden fordelingen av utmattingslevetid er ukjent, anbefales det at kurven ikke ekstrapoleres utenfor intervallet man tester.

Videre anbefales det at utmattingslevetiden ved en spesifikk spenningsamplitude ikke estimeres med mindre enn 5 prosent usikkerhet da man kun har tilnærmet seg S-N forholdet med den rette linjen, dvs. at man kun har de estimerte parametrene for linjen, ikke linjen i seg selv.

Standarden spesifiserer at levetiden, N, er den avhengige variabelen (responsen), og spenningen, σ, er den uavhengige variabelen (forklaringen) og viser til at aksebruken ikke følger ordinær praksis for plotting. Se kapittel 3.3 for utdypning.

---

1 Se standarden for fullstendig oversikt over begrensninger.

ASTM setter følgende krav til ulike typer tester:

Tabell 4: Minstekrav til antall prøvestaver og replikasjonverdier

Type	Antall staver	Replikasjon
Preliminære undersøkelser	6-12	17-33
Utvikling av komponenter	6-12	33-55
Konstruksjondata	12-24	50-75
Pålitlighetdata	12-24	75-88

Replikasjon for testen angis ved:

$$r = 100\left(1 - \frac{l}{n}\right) \tag{1}$$

Dette betyr at det er ønskelig å konsentrere antall prøver på et begrenset antall spenningsnivåer.

4.1.3. Statistisk analyse

ASTM setter følgende krav til testdataen:

a. Levetidsdataen kommer fra tilfeldige, uavhengige prøver.
b. Det er ikke "run-outs" eller avbrutte tester i hele belastningintervallet.
c. S-N forholdet kan beskrives med en lineær modell.
d. Den logaritmiske normalfordelingen beskriver levetiden N.
e. Variansen av log-normalfordelingen er konstant.

Hvis dette oppfylles kan man estimere linjen med standard lineær regresjonsanalyse. Dette utdypes nærmere i kapittel 5.

Det spesifiseres at konfidensintervallene for α og β er forholdsvis nøyaktige selv om krav (d) ikke oppfylles fullstendig pga. robustheten til t-fordelingen brukt i intervallet.

4.1.4. Designlinje

ASTM tar ikke direkte for seg retningslinjer for beregning av linje for design, og går kun inn på beregning av konfidensområder til modellen. Denne formelen likner på utgangspunktet for designlinjene, men tar kun for seg eksisterende data. Formler beregnet for å estimere forventede verdier for nye prøver har et tilleggsledd, så formelen for konfidensområdet kan derfor ikke brukes.

4.2. DNV-RP-C203 [6]

Standarden tar for seg generelt design mot utmatting av offshore stålstrukturer, og inneholder ett vedlegg (D.7, side 154) som tar for seg S-N kurver og beregning av disse.

4.2.1. Antagelser/Begrensninger[1]

Standarden er gyldig for stålmaterialer i luft med flytgrense under 960 MPa og stålmaterialer i sjøvann opp til 550 MPa. Videre er standarden beregnet på utmatting fra høysykel belastning, dvs. mer enn 10^4 sykler. Vanlige testdata for S-N kurver oppgis til å ligge mellom 10^4 og $5{\cdot}10^6$ sykler.

Det antas at det holder å vurdere spenningsvidden for å beregne utmattingslevetid, dvs. at middelspenningen ignoreres.

4.2.2. Anbefalinger

For nye typer forbindelser anbefales det å teste minst 15 prøver for å etablere en S-N kurve. Disse bør fordeles på minst tre ulike spenningsvidder innenfor den relevante S-N regionen slik at man kan bestemme en representativ stigning.

Videre spesifiserer DNV viktigheten av ingeniørvurderinger både under testing og ved vurdering av S-N kurven for reelle strukturer når man har brukt småskala prøvebiter under testing.

4.2.3. Statistisk analyse

DNV henviser til International Institute of Welding sitt dokument IIW-XIII-WG1-114-03, samt ISO 12107 for statistisk vurdering av testdataen.

4.2.4. Designlinje

Designkurven skal gi en 97,7 % sannsynlighet for overlevelse og kan beregnes som snittlinjen fra måledataen (regresjonslinjen) minus to standardavvik, hvor man antar at testdataen er gaussfordelt (normalfordelt) ved logaritmisk form.

Når man har et begrenset antall tester må man ta hensyn til statistisk usikkerhet blant testdataen. DNV setter krav om at snittkurven beregnes med minst 75 % konfidens og oppgir en oversikt over korreksjonsfaktorer for designkurven basert på antall tester. Korreksjonsfaktoren tilsvarer antall standardavvik som trekkes fra regresjonslinjen.

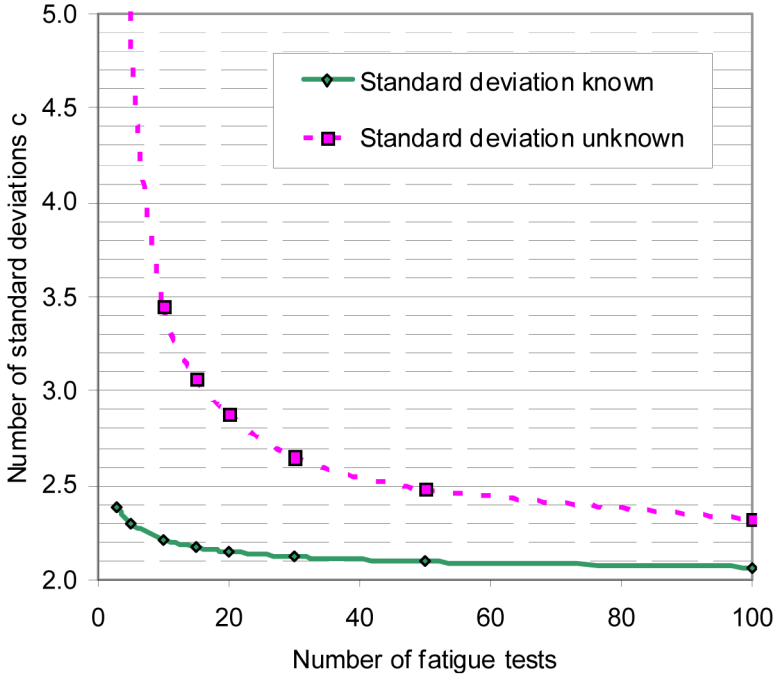
1 *Se standarden for fullstendig oversikt over begrensninger.*

Tabell 5: Antall standardavvik man skal trekke fra for 97.7 % sannsynlighet for overlevelse [6].

Antall tester	Kjent S	Ukjent S
3	2.39	9.24
5	2.30	5.01
10	2.21	3.45
15	2.17	3.07
20	2.15	2.88
30	2.12	2.65
50	2.10	2.48
100	2.07	2.32
∞	2.00	2.00

DNV skiller mellom en ren statistisk fremgangsmåte og en ingeniørbasert fremgangsmåte. Skal man følge den førstnevnte må man anse standardavviket som ukjent da man regner den ut fra måledataen. Dette gjør at man må legge inn en stor sikkerhet i designlinjen med mindre man har utført et stort antall tester.

Med en ingeniørbasert fremgangsmåte kan man vurdere standardavviket som kjent hvis man har utført liknende tester tidligere med samme standardavvik. Dette gir en mye mindre korreksjonsfaktor ved et lavt antall prøver.



Figur 10: Korreksjonsfaktor ved kjent og ukjent standardavvik [6].

Som man kan se av Figur 10 vil disse verdiene nærme seg hverandre, men det krever et stort antall tester og er ofte lite gjennomførbart på grunn av tids og kostnadsaspekter.

Det er verdt å merke seg at man kan forvente et større standardavvik for komplekse overganger og det kan derfor lønne seg å kontrollere at dataen ser homogen ut og at spredningen ikke er større enn det man vanligvis observerer ved testing.

4.3. BS 7608:2014 [7]

BS 7608 er en generell standard for utmattingsdesign og vurdering av stålprodukter, og er gjeldende for alle områder der det ikke er en egen spesialisert standard.

4.3.1. Antagelser/Begrensninger[1]

Standarden gjelder når den beregnede største fiberpåkjenningen på nettoarealet ikke overstiger 60 % av flytgrensen ved normal drift og 80 % under ekstreme lastpåkjenninger. Disse verdiene gjelder utenfor områder med spenningskonsentrasjoner.

Standarden er gyldig for stålmaterialer med en flytgrense mellom 200 MPa og 960 MPa og bruddgrense mellom 360 MPa og 1 200 MPa hvor materialet har en tykkelse minst 3 mm.

4.3.2. Anbefalinger

Ved testing av sveisede komponenter anbefales det å teste fullskala prøver, men det gis også tips til korreksjoner hvis dette ikke er mulig.

Videre anbefales det å velge spenningsvidder forholdsvis jevnt fordelt over den lineære delen av kurven (vanligvis 10^5 til $2{\cdot}10^6$ sykler), med flere prøver på noe av nivåene. De setter også som et alternativ å kjøre alle testene på samme spenningsnivå for noen typer valideringstester som ikke gjennomgås her.

BS setter 8 prøver som et minimum antall prøver for testing, men det er verdt å merke seg at dette er tester for å validere bruk av kurver, ikke design av nye.

Videre anbefales det å simulere bruksmiljøet under utmattingstesting hvis det skiller seg fra testmiljøet. Da bør også lastfrekvensen tilpasses forventet bruksfrekvens.

4.3.3. Statistisk analyse

De statistiske metodene følger ISO 12107 og selve utregningen av linjen utdypes ikke i større grad. Standarden fokuserer på å validere bruk av en eksisterende designkurver ved hjelp av testing fremfor å lage egne kurver.

Under avsnittet om å produsere nye kurver henviser standarden til regresjonsanalyse, men trekker også her inn bruk av eksisterende kurver, da det forventes at den nye modellen skal ha samme stigning *m* som designkurven man bruker som utgangspunkt.

Da dette følger en litt annen metodikk en hva oppgaven benytter seg av, vil ikke disse testene benyttes her.

---

1 Se standarden for fullstendig oversikt over begrensninger.

4.3.4. Designlinje

Britisk standard setter designlinjen 2 standardavvik under snittlinjen, men det åpnes opp for å bruke et mindre avvik der et brudd ikke vil ha store konsekvenser, eller der brudd enkelt kan lokaliseres og repareres. De oppgir følgende tabell:

Tabell 6: Sannsynlighet for brudd (side 60 i standarden).

Nominell sannsynlighet for brudd i %	Standardavvik
50	0
31	0,5
16	1,0
2,3	2,0
0,14	3,0

0 standardavvik vil da tilsvare regresjonslinja og 2 standardavvik tilsvarer vanlig designlinje. Disse verdiene er ikke korrigert for antall prøver.

4.4. ISO 12107 [8]

ISO 12107 tar for seg utmattingstesting, statistisk planlegging og analyse av data for metalliske materialer. Standarden tar for seg både vurdering av utmattelsesliv ved en gitt spenning og vurdering av utmattingsstyrke ved ett gitt liv. I denne oversikten vil det være fokus på vurdering av utmattelsesliv.

Det er verdt å merke seg at standarden bruker X og Y motsatt av andre standarder på området, noe som det er korrigert for i gjennomgangen under.

4.4.1. Antagelser/Begrensninger[1]

Standarden dekker kun analyse av materialer som svikter grunnet en enkelt bruddmekanisme, og tar ikke for seg mer komplekse systemer.

Standarden forholder seg heller ikke til ”runouts” eller ukomplette datasett.

Videre antar standarden at levetiden er logaritmisk normalfordelt, selv om de setter opp Weibull fordeling som ett alternativ.

4.4.2. Anbefalinger

Standarden spesifiserer viktigheten av et randomisert utvalg av prøvestaver og oppgir noen retningslinjer for dette.

Ved levetidstesting anbefaler ISO minst 28 prøvestaver for pålitlighetsdata, men kun 7 staver for preliminære (foreløpige) undersøkelser. Ved testing av utmattingsstyrke stilles det ett høyere krav til

1 *Se standarden for fullstendig oversikt over begrensninger.*

antall prøvestaver.

ISO kommer videre med en liste over hva en testrapport bør inneholde, noe som vil bli tatt hensyn til i utvikling av resultater fra dataprogrammet.

4.4.3. Statistisk analyse

ISO legger stort fokus på visuell vurdering av resultatene. Måledataen skal fremstå som en rett linje når de plottes med en dobbel-logatitmisk skala. Hvis en eller flere punkter avviker mye fra den rette linjen er det ofte et resultat av ugyldig data. Resultatene må derfor undersøkes nøye for å avgjøre om man kan se bort i fra de aktuelle punktene. Alternativt kan man vurdere en annen fordeling for måledataene, f.eks. Weibull.

Videre anbefaler standarden å dele opp måledataen hvis det er tydelig at de støtter to distinktivt ulike bruddmekanismer og at man vurderer de to gruppene separat. Et eksempel kan være hvis man har sprekkinitiering både fra overflate og fra innvendige uregelmessigheter.

Når det kommer til beregning av selve S-N kurven tar standarden for seg både lineær og kurvet lineær respons og stiller derfor med mer komplekse formler for kurven hvor man bruker matriseregning for å finne verdiene for linjen. Det spesifiseres at disse ikke lar seg løse med minste kvadraters metode, som er vanlig metode for lineær regresjon, og ISO henviser til eksterne kilder for denne typen beregninger. For rene lineære sammenhenger oppgir de derimot ett sett med formler basert på vanlig regresjonsanalyse, noe som utdypes nærmere i kapittel 5.

4.4.4. Designlinje

Standarden oppgir en formel for nedre toleranselinje, her kalt designlinjen, basert på en ikkesentral Student t-fordeling. Denne tar hensyn til antall prøvestaver og er avhengig av spenningsvidden der verdien regnes ut. Dette gjør at linjen ikke blir lineær, noe som utdypes under kapittel 5.

5. Statistisk evaluering av testdata

Dette kapittelet vil ta for seg statistikken for evaluering av testdata basert på formelverk fra standardene og statistisk litteratur. Formler og uttrykk er skrevet om så de følger symbolbruk og terminologi gitt i kapittel 1.5.

Det antas at sammenhengen mellom spenningsvidde og antall sykler til brudd kan fremstilles som en rett linje i et dobbel-logaritmisk diagram, hvor man har en stigning β for log(Δσ) og et krysningspunkt med log(N) aksen ved α [5].

$$\log N = \alpha + \beta \cdot \log(\Delta\sigma) \tag{2}$$

Man ønsker gjerne å forholde seg til spenningen og antall sykler direkte og skriver gjerne om funksjonen:

$$\alpha = \log c \tag{3}$$

$$\beta = -m \tag{4}$$

$$\log N = \log c - m \cdot \log(\Delta\sigma) \tag{5}$$

(5) kan da skrives om til formen:

$$N = c \cdot \Delta\sigma^{-m} \tag{6}$$

5.1. Beregne regresjonslinjen og varians

Verdiene for α og β finner man ved lineær regresjonsanalyse ved å bruke minste kvadraters metode [5, 9]. Her bruker alle standardene samme formel, selv om formen er skrevet om for enklere håndregning i noen tilfeller:

$$Y = \alpha + \beta X \tag{7}$$

$$\hat{\alpha} = \bar{Y} - \hat{\beta}\bar{X} \tag{8}$$

$$\hat{\beta} = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^{n}(X_i - \bar{X})^2} \tag{9}$$

$$X_i = \log \sigma_i \tag{10}$$

$$Y_i = \log N_i \tag{11}$$

$$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n} \tag{12}$$

$$\bar{Y} = \frac{\sum_{i=1}^{n} Y_i}{n} \tag{13}$$

Variansen ved normalfordelt log(N) er gitt ved:

$$\hat{S}^2 = \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{n-2} \tag{14}$$

$$\hat{Y}_i = \hat{\alpha} + \hat{\beta} X_i \tag{15}$$

5.2. Beregne konfidensintervall og konfidensbånd

Man kan regne ut konfidensintervall for α og β (som kan gjøres om til konfidensintervall for c og m ved hjelp av formel (3) og (4)) med:

$$\hat{\alpha} \pm t_{p/2,n-2} \cdot \hat{S} \cdot \sqrt{\frac{1}{n} + \frac{\bar{X}^2}{\sum_{i=1}^{n}(X_i - \bar{X})^2}} \tag{16}$$

$$\hat{\beta} \pm t_{p/2,n-2} \cdot \hat{S} \cdot \frac{1}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}} \tag{17}$$

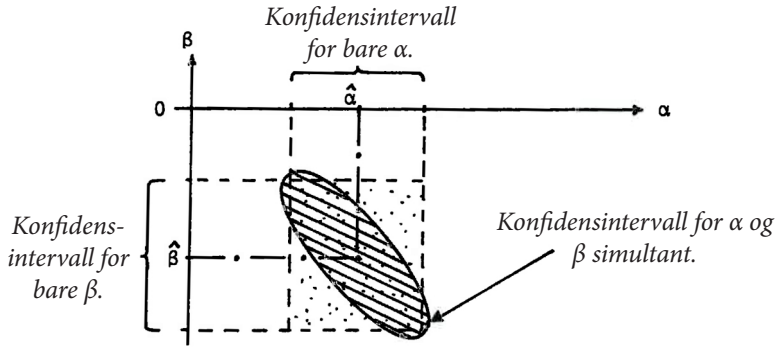
”p” er usikkerheten man godtar, f.eks. 0.05 for 5 prosent usikkerhet, noe som gir ett 95 % konfidensintervall. Det er ikke anbefalt å bruke et høyere konfidensintervall enn 95 % siden man jobber med en tilnærmet lineær modell [5].

Det er viktig å merke seg at et 95 % konfidensintervall betyr at man ved en serie av analyser kan

forvente å få en verdi som ligger innenfor intervallet i 95 % av tilfellene. Det betyr ikke at verdien vil falle innenfor intervallet med 95 % sannsynlighet. Sistnevnte går over på prediksjon.

α og β er avhengige av hverandre, og begge kan derfor ikke velges fritt. I rapportene fra Sintef [9, 10] bruker de en konfidensregion som tilsvarer en ellipse sentrert ved $(\hat{\alpha}, \hat{\beta})$ for å illustrere dette. Konturen til denne regionen er gitt ved:

$$n \cdot (\hat{\alpha} - \alpha)^2 + 2 \cdot (\hat{\alpha} - \alpha)(\hat{\beta} - \beta) \cdot \sum_{i=1}^{n} X_i + (\hat{\beta} - \beta)^2 \cdot \sum_{i=1}^{n} {X_i}^2 = 2 \cdot \hat{S}^2 \cdot f_{p,2,n-2} \quad (18)$$



Figur 11: Simultant konfidensområde for α og β [10].

Man ser her at det sammenfallende konfidensområdet for begge verdiene er mye mindre enn det rektangulære området som representerer de individuelle konfidensintervallene.

Konturformelen kan brukes for å finne konfidensintervall for α eller β når en av verdiene er gitt.

Ørjaseter har tatt hensyn til dette i sin rapport [10] og har oppgitt alternative konfidensintervall for α og β hvor variasjonen er begrenset til tyngdepunktet av dataen. Dette forenkler formlene og gir:

$$\hat{\alpha} \pm \hat{S} \cdot \sqrt{\frac{2f_{p,2,n-2}}{n}} \quad (19)$$

$$\hat{\beta} \pm \hat{S} \cdot \sqrt{\frac{2f_{p,2,n-2}}{\sum_{i=1}^{n} {X_i}^2}} \quad (20)$$

Disse konfidensintervallene vil ikke bli brukt i programmet ettersom man ikke ønsker å gjøre forenklinger på brukerens vegne.

Ved hjelp av konfidensintervallene for α og β kan konfidensbåndet for hele regresjonslinjen gis ved:

$$\hat{\alpha} + \hat{\beta}X \pm \sqrt{2f_{p,2,n-2}} \cdot \hat{S} \cdot \sqrt{\frac{1}{n} + \frac{(X - \bar{X})^2}{\sum_{i=1}^{n}(X_i - \bar{X})^2}} \quad (21)$$

Denne linjen gir kun et konfidensområde for de målte verdiene og kan ikke brukes for å estimere fremtidige verdier.

5.3. Test av gyldighet

ISO benytter korrelasjonskoeffisienten r^2 for å beskrive kvaliteten til modellen. Her vil en høy verdi bety at modellen forklarer variasjonen i datasettet godt, og man ønsker en verdi på 0,9 eller bedre [8].

$$r^2 = \frac{(\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y}))^2}{\sum_{i=1}^{n}(X_i - \bar{X})^2 \cdot \sum_{i=1}^{n}(Y_i - \bar{Y})^2} \tag{22}$$

ASTM anbefaler at man ikke bruker denne verdien for å vurdere kvaliteten til linjen, og gir eksempler på hvordan testsett med veldig ulikt resultat fra testing for linearitet gir forholdsvis lik korrelasjonskoeffisient. En svakhet med r^2 er at den ikke tar hensyn til antall punkter, slik at en modell med få verdier vil kunne få bedre r^2 verdi enn en tryggere modell med flere punkter, men også mulighet for større variasjon.

Siden noen brukere kan være vant med å bruke r^2 vil den allikevel bli lagt inn i programmet, men vil komplementeres med andre tester.

ASTM oppgir en test for å kontrollere at den lineære modellen er tilstrekkelig:

$$F = \frac{\sum_{i=1}^{l} k_i\left(\hat{Y}_i - \bar{Y}_i\right)^2/(l-2)}{\sum_{i=1}^{l}\sum_{j=1}^{k_i}\left(Y_{ij} - \bar{Y}_i\right)^2/(n-l)} \leq F_{p,l-2,n-l} \tag{23}$$

hvor l er antall spenningsnivåer, k_i er antall staver på hvert nivå (replikater) og n er totalt antall prøvestaver. Modellen forutsetter et minimum av 3 spenningsnivåer.

Her tester man om hypotesen for linearitet er gyldig (null-hypotese), og hypotese må forkastes dersom F-verdien overstiger fisher-verdien med ønsket sikkerhet ved de gitte frihetsgradene. F-testen sammenligner variansen av gjennomsnittsverdiene rundt snittlinjen mot variansene innad blant replikatverdiene. Siden den sistnevnte er uavhengig av modellen vil man kunne teste om modellen er lineær, dvs. om den lineære modellen gir en god forklaring.

Hvis den lineære modellen må forkaster anbefaler ASTM at man vurderer en ikkelineær modell.

IIW [11] anbefaler visuell vurdering av resultatene for å kontrollere at selve plottet av resultatene følger en lineær form. De anbefaler også at residualplottet vurderes for å kontrollere jevn spredning av avvik. Finner man tydelige gjenkjennbare mønstre i plottet kan det tyde på at modellen enten ikke er lineær, og at man bør vurdere en kvadratisk modell, eller at modellen ikke er statistisk uavhengig. Rapporten oppgir ikke formler for vurdering av dataen, men henviser til statistisk programvare og navngir noen statistiske tester.

ISO [8] tar også for seg viktigheten av å vurdere residualplottene og definerer residualene som:

$$e_i = Y_i - \hat{Y}_i \tag{24}$$

hvor estimatet av Y er gitt i formel (15). Standarden oppgir residualene som avvik i spenning, men da modellen tar for seg testing for både spenning og testing for levetid, er avvik fra levetid brukt her. Dette samsvarer også med statistisk litteratur[12], da levetiden er responsvariabelen.

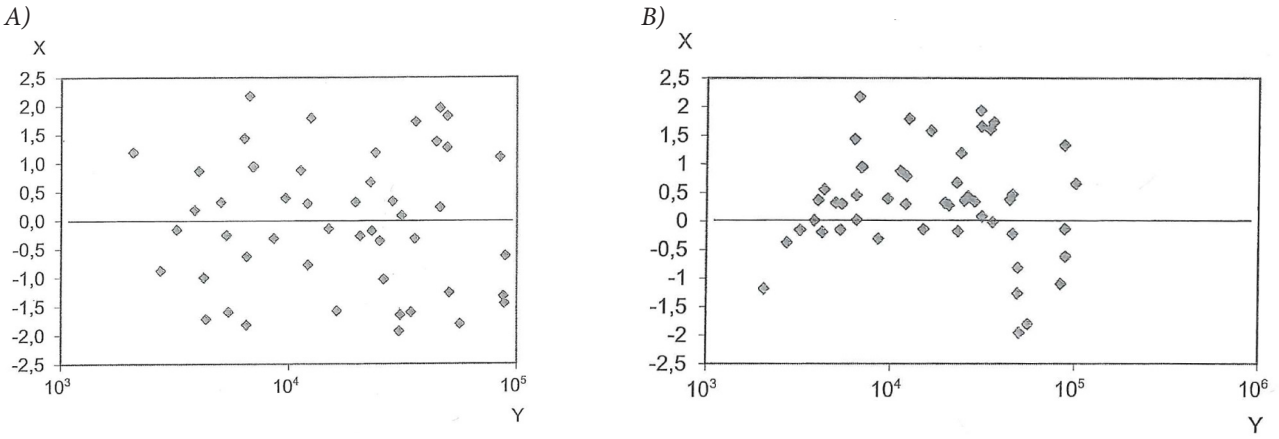
Summen av residualer skal alltid bli null og de bør være sentrert om 0 på y-aksen. ISO har videre skalert y-aksen etter standardavviket og plotter dermed de standardiserte residualene.

Formel for det standardiserte residualplottet kan man hente fra statistikklitteratur [12] og de korrigerte enkeltverdiene for plottet gis ved:

$$d_i = \frac{e_i}{\hat{S}} \tag{25}$$

hvor standardavviket, $\hat{S}$, er roten av formel (14). Residualene plottes mot estimert logaritmisk levetid.

Ved å vekte verdiene mot standardavviket kan man bruke egenskapene til normalfordelingen (se Figur 15) til å vurdere spredningen. Man forventer da at ca. 68 % av residualene havner innenfor ett standardavvik (som nå får verdien ”1” og ”-1” på X-aksen), og rundt 95 % innenfor 2 standardavvik.



Figur 12: Eksempel på residualplott med god spredning (A) og med spredning som antyder behov for en kvadratisk modell(B)[8]. X- standardiserte residualer. Y- estimert liv.

ISO anbefaler videre at man plotter et normal-sannsynlighetsplot, hvor man kan evaluere om resultatene virkelig er normalfordelt, og dermed følger en forholdsvis rett linje. Siden modellen er basert på at dataen er normalfordelt er det viktig å kontrollere at dette stemmer.

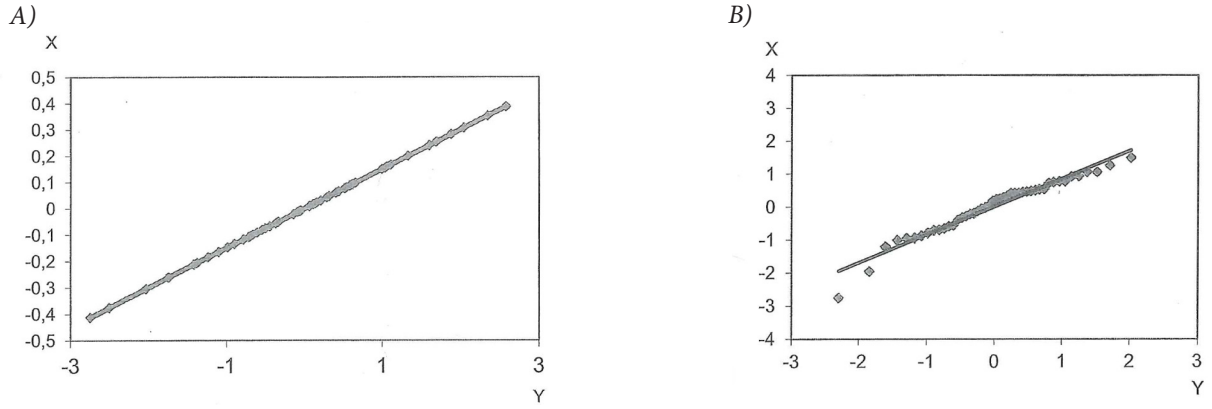
Bakgrunnen for et normal-sannsynlighetsplott finner man også i statistikklitteratur [12]. Utgangspunktet for dette plottet er at man sorterer residualene i økende størrelse og plotter mot jevnt fordelte punkter. Man bruker da enten den kumulative sannsynligheten beregnet med formel (26),

$$P_i = \frac{\left(i - \frac{1}{2}\right)}{n} \tag{26}$$

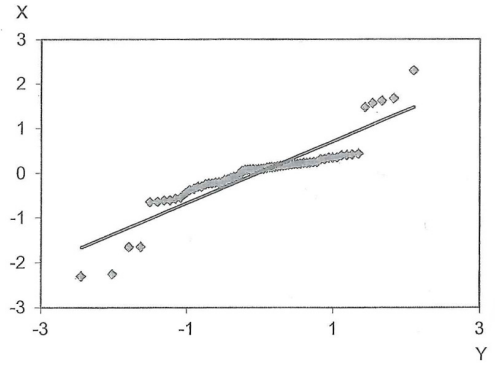
hvor i = 1, 2, ..., n, eller mer vanlig: P_i verdiene gjort om til standardavvik ved hjelp av egenskapene til normalfordelingen. Disse verdiene hentes ut fra tabell eller med programvare. Den førstnevnte metoden vil gi en skala fra 0 til 1, mens den sistnevnte gir verdier fra ca. -3 til 3 på den horisontale aksen. Denne sammenhengen er illustrert i Figur 15.

Den kumulative fordelingen av residualpunktene skal ligge omtrentlig på en rett linje hvis dataen er normalfordelt. Denne vurderingen gjøres visuelt hvor man fokuserer på de sentrale verdiene på grafen. En svakhet med denne metoden er at man trenger mange målepunkter, vanligvis rundt 20 [12], for enkelt å vurdere resultatene grafisk.

I Figur 13 og Figur 14 kan man se eksempler på gode og mindre gode plot.



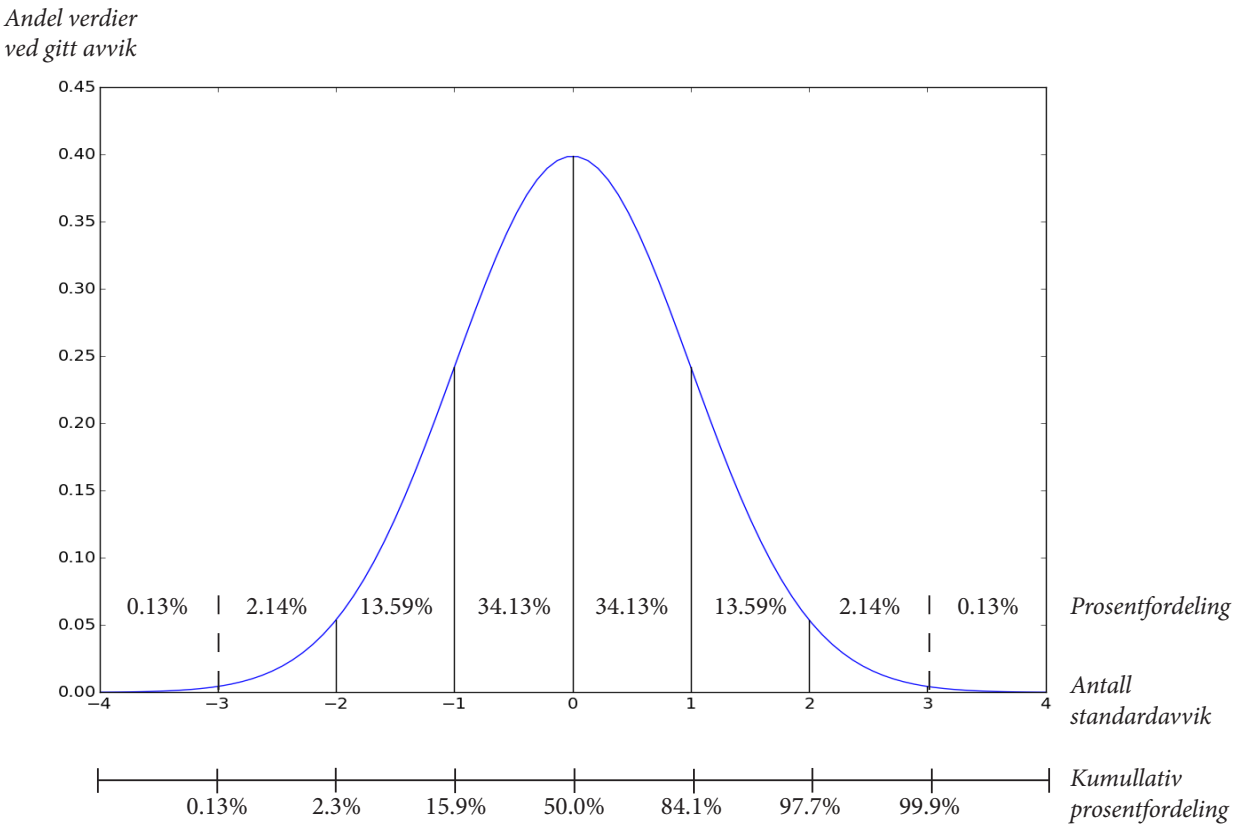
Figur 13: Eksempel på normal-sannsynlighetsplot med god (A) og akseptabel (B) konformitet for normalfordeling[8]. X- standardiserte residualer. Y- normal sannsynlighet.



Figur 14: Eksempel på normal-sannsynlighetsplot med marginalt akseptabelt konformitet for normalfordeling [8]. X- standardiserte residualer. Y- normal sannsynlighet.

5.4. Beregning av designlinjen

Ved beregning av designlinjen tar man utgangspunkt i egenskapene til en standard normalfordeling (se Figur 15). Man ser at størsteparten av verdiene befinner seg rundt midten (regresjonslinjen i vårt tilfelle), og det er et lavt antall verdier som forventes å falle mer enn 2 standardavvik fra snittet. Siden man antar at prøvedataen er standard normalfordelt kan man bruke denne egenskapen til å regne ut en designlinje med ønsket sikkerhet mot brudd.



Figur 15: Egenskaper til standard normalfordeling.

Britisk Standard (4.3) setter designlinjen 2 standardavvik under regresjonslinjen for dermed å få med en stor andel av variasjonen innad i prøvene. Siden det kun er avvikene på nedsiden av linjen som er av betydning (en høyere levetid vil ikke være ett problem), vil man kunne anslå at 97,7 % av verdiene vil falle på "trygg side" av linjen. En svakhet med denne metoden er at den ikke tar hensyn til hvor sikker linjen er ut i fra antall prøvestaver som er lagt til grunn. Spesielt ved få prøvestaver vil man kunne få store usikkerhetsmomenter.

DNV (4.2) tar hensyn til antall prøver linjen baseres på og setter avviket fra 2 standardavvik og oppover (se Tabell 5). Ved få prøver får man store sikkerhetsmarginer, og DNV tar også hensyn til hvor mange parametere som anslås fra prøvedataen.

Både designlinjen fra Britisk Standard og DNV kan beregnes med formelen nedenfor:

$$\log N_{Design} = \log N - b \cdot \hat{S} \tag{27}$$

$$\log N_{Design} = \log C - b \cdot \hat{S} - m \cdot log(\Delta\sigma) = \log \bar{C} - m \cdot log(\Delta\sigma) \tag{28}$$

$$\log \bar{C} = \log C - b \cdot \hat{S} \tag{29}$$

Her er b antall standardavvik man trekker fra, som for DNV hentes fra tabell. Som man ser fra formlene over vil designlinjen tilsvare en vanlig regresjonslinje med korrigert verdi for skjæringspunktet C. Linjen vil ha samme stigningstall, men ligge b standardavvik under regresjonslinjen.

ISO oppgir en formel for designlinjen basert på en ikkesentral Student t-fordeling og korrigert for antall prøvestaver:

$$\log N_{1-P,1-p} = \log N - k_{1-P,1-p,n-2} \cdot \hat{S} \cdot \sqrt{\frac{n+1}{n} + \frac{(X-\bar{X})^2}{\sum_{i=1}^{n}(X_i-\bar{X})^2}} \tag{30}$$

Da k-verdiene må regnes ut stiller ISO med en tabell med disse verdiene. Man velger da k-verdi ut i fra antall prøver, påliteligheten til prediksjonen 1-P og konfidensnivået til prediksjonen 1-p. I tabellen oppgir de verdier for konfidensnivået (1-p) på 90 og 95 % og verdier for sannsynligheten P på 10, 5, 1 og 0,1 %. Da 0,1 % gir en veldig konservativ linje er den ikke brukt i programmet.

Ørjaseter [10] og IIW [11] viser til en liknende funksjon, men bruker vanlig Student t-fordeling:

$$\log N_{1-p} = \log N - t_{\frac{p}{2},n-2} \cdot \hat{S} \cdot \sqrt{\frac{n+1}{n} + \frac{(X-\bar{X})^2}{\sum_{i=1}^{n}(X_i-\bar{X})^2}} \tag{31}$$

IIW nevner også en modell med ikkesentral t-fordeling, da denne er mindre følsom for antall prøver, som har likhetstrekk med ISO sin modell. Da den ikke er lik ISO sin modell, og Ørjaseter ikke fant noen store forskjeller fra den enklere metoden med vanlig Student t-fordeling i sin rapport [10], er den ikke tatt med her.

Ørjaseter påpeker at ved å anta at spredningen tilsvarer det man har i tyngdepunktet for nåværende testdata, får man en forenklet formel som gir en rett linje:

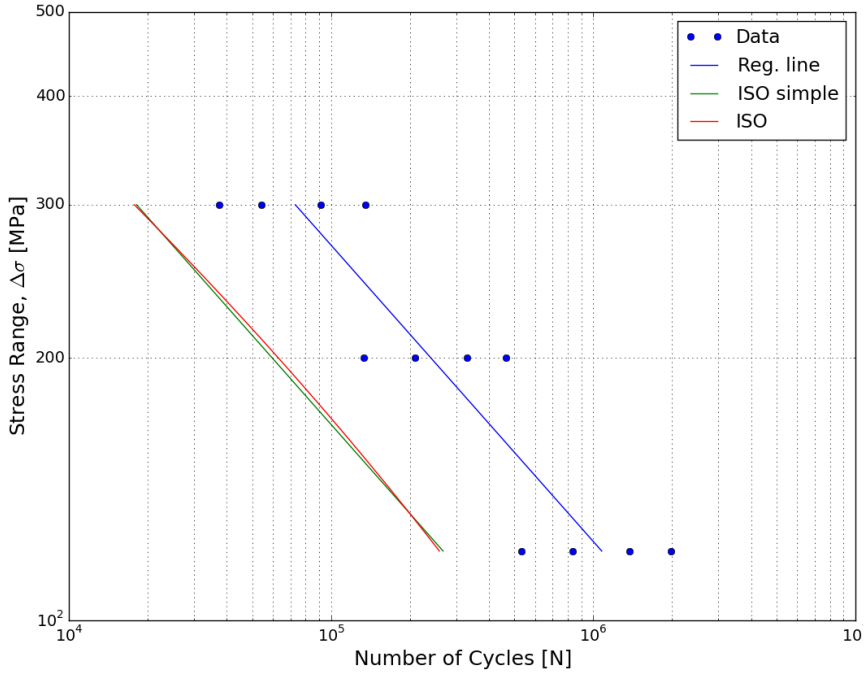
$$\log N_{1-p} = \log N - t_{\frac{p}{2},n-2} \cdot \hat{S} \cdot \sqrt{\frac{n+1}{n}} \tag{32}$$

IIW åpner også for den samme forenklingen ved å si at det siste leddet i formel (31) kan ignoreres hvis man antar at designkurven kun vil brukes for spenningsverdier nær snittet. ISO foreslår ikke tilsvarende forenklinger. Dette gjør bruken av designkurven deres krevende, da man ikke får en vanlig formel for linjen. Det er derfor naturlig å vurdere samme forenklingen:

$$\log N_{1-P,1-p} = \log N - k_{1-P,1-p,n-2} \cdot \hat{S} \cdot \sqrt{\frac{n+1}{n}} \tag{33}$$

Ser man på Figur 16, hvor formel (30) er plottet mot den forenklede formelen (33) på ett sett eksempelverdier med stor spredning, ser man at linjene ikke avviker i stor grad. Den forenklede linjen får en økning i sikkerhet ved sentrale verdier, mens den ved ytterpunktene er så vidt mindre sikker.

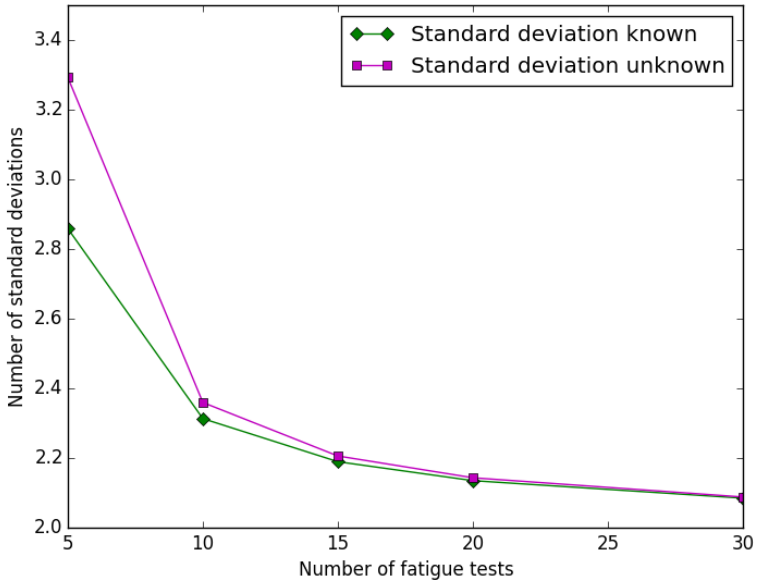
Siden det er små forskjeller mellom linjene, og da hovedsakelig på trygg side av linjen, er den forenklede modellen brukt videre, da den gir mye større bruksområde.



Figur 16: Sammenlikning av vanlig designlinje fra ISO mot forenklet modell.

Rapporten til IIW inneholder også flere forenklinger for modellen: for mer enn 20 prøver viser de til at det første leddet under rottegnet kan settes lik 1 og har man mer enn 40 prøver kan man erstatte t-verdien med 2. Videre anbefaler de at man for færre enn 10 prøver bør øke frihetsgradene til Student t-fordelingen med 1. Disse forenklingene er ikke implementert.

Ser man nærmere på modellen som bl.a. IIW bruker, ser man at den er avhengig av antall frihetsgrader i Student t-fordelingen. Det har sammenheng med antall parametere som estimeres ut i fra prøvedataen. Bruker man litt av den samme logikken som DNV bruker, kan man få to ulike korreksjonsfaktorer for designlinjen, avhengig av om man estimerer standardavviket eller ikke.



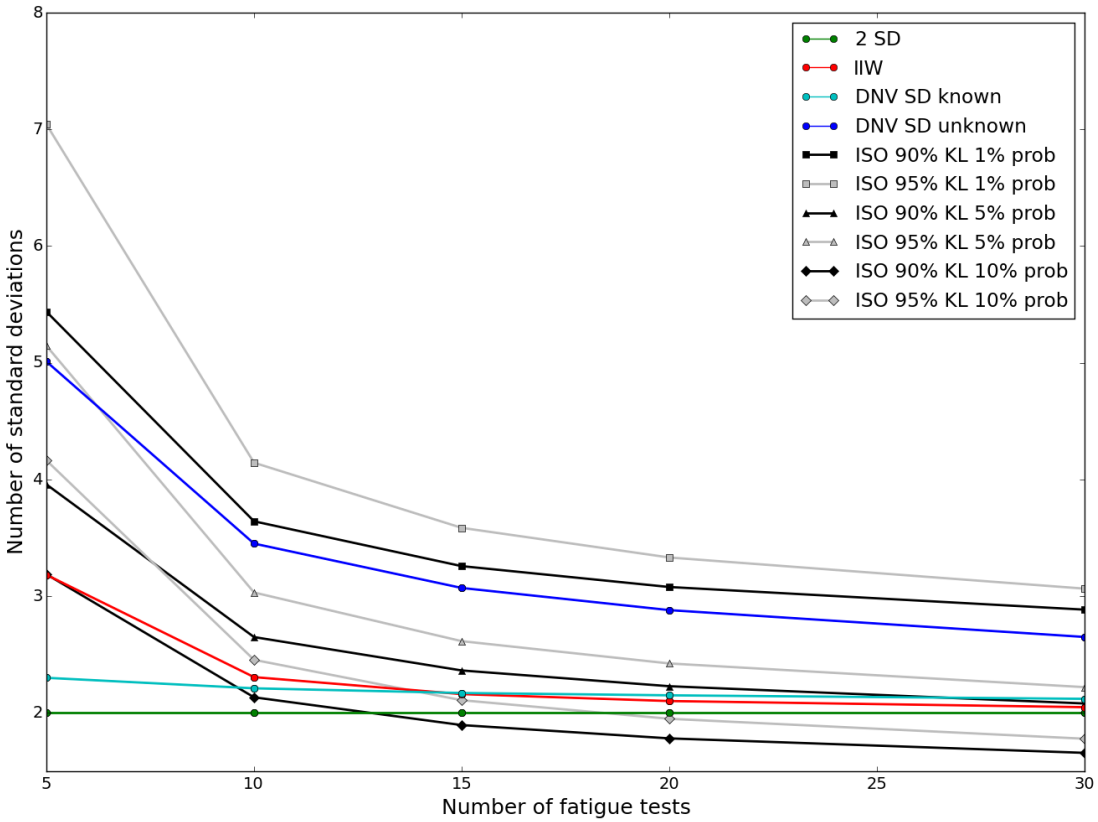
Figur 17: Korreksjonsfaktor (IIW) for designlinje basert på antall frihetsgrader

Her ser man at man får en forholdsvis stor forskjell i korreksjonsfaktor ved få verdier, mens det for

flere prøver ikke blir merkbar forskjell. Dette forklarer ikke helt de store og vedvarende forskjellene i verdiene hos DNV, selv om frihetsgrader trolig har en innvirkning.

Da endring av frihetsgrader ikke omfattes av litteraturen som oppgir formelen, er ikke dette implementert videre i programmet. Muligheten for en mindre korreksjonsfaktor hvis man kjenner standardavviket får man uansett med DNV sin modell.

Sammenlikner man korreksjonsfaktorene til de ulike modellene får man følgende oversikt:



Figur 18: Oversikt over korreksjonsfaktorer fra ulike modeller ved antall tester.

Spesielt ISO sine korreksjonsfaktorer er interessante å vurdere her. Man ser at en 1 % sannsynlighet gir en veldig stor grad av sikkerhet. 90 % konfidensnivå (KL), 1 % sannsynlighet gir linjen nærmest DNV sin sikreste linje, men ISO har her enda større sikkerhetsmargin. 90 % KL, 5 % sannsynlighet er den linjen som minner mest om de andre designlinjene. Den er derimot betydelig mer konservativ ved få verdier. Begge 10 % sannsynlighetslinjene starter forholdsvis høyt, men går etter hvert under 2 standardavvik i verdi. Dette betyr at man her får en sikkerhet som tilsvarer mindre enn 97,7 % grensen de andre standardene jobber med.

Ser man på de andre korreksjonsfaktorene ser man at de fleste nærmer seg 2. Mens DNV sin linje med kjent standardavvik hele tiden holder seg nærme, starter IIW sin høyere oppe, men går til gjengjeld under DNV sin linje ved et større antall prøver.

5.5. Sammenlikning av to datasett

Man finner statistiske formler for å sammenlikne to datasett hos IIW [11] og i begge rapportene til Sintef [9, 10]. Da rapportene til Sintef, spesielt [9], tar for seg mer av metodikken bak formlene, er disse brukt som utgangspunkt. Formelsettet til IIW er også mindre tydelig da de bl.a. kun oppgir frihetsgrader for tester utført på samme spenningsnivå.

Det er flere metoder for å bestemme om stigning og skjæringspunkt til aksen kan antas like hver for seg og testene kan utledes til formler som enkelt lar seg løse. For å teste for begge parameterne samtidig må man derimot gjennomføre en litt mer omfattende utregning. Dette kan bl.a. gjøres med matriseregning og man kan da velge hvor mange parametere man vil teste for. Dette må hentes fra ekstern litteratur da kun oppsett av F-testen ved hjelp av de kvadrerte residualene oppgis.

Siden den førstnevnte metoden er enklere i bruk, mens metoden med matriseregning gir flere resultater, vil begge metodene gjennomgås. I programmet vil metoden med matriseregning benyttes da den kan brukes for alle testene for α og β, og den ikke bruker forenklingen med felles varians.

5.5.1. Teste for felles varians

Variansen til linjen vil ha stor betydning for designlinjen og det er derfor av stor interesse å teste om man kan anta felles varians mellom linjene. Man tester da hypotesen H_0: $S_1^2 = S_2^2$, mot H_1: $S_1^2 \neq S_2^2$ og forkaster H_0 hvis en av de følgende er oppfylt:

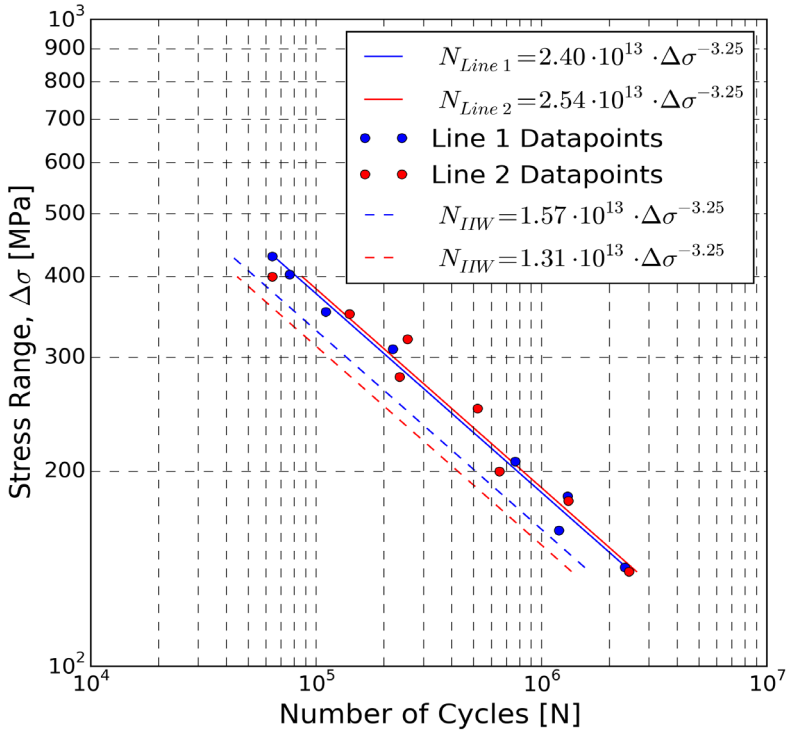
$$\frac{\hat{s}_1^2}{\hat{s}_2^2} > f_{\frac{p}{2},n_1-2,n_2-2} \tag{34}$$

$$\frac{\hat{s}_1^2}{\hat{s}_2^2} < \frac{1}{f_{\frac{p}{2},n_1-2,n_2-2}} \tag{35}$$

Man kan regne ut felles varians med følgende formel:

$$\hat{s}^2 = \frac{(n_1 - 2) \cdot \hat{s}_1^2 + (n_2 - 2) \cdot \hat{s}_2^2}{n_1 + n_2 - 4} \tag{36}$$

Nedenfor ser man ett eksempel på hvordan Linje 2, som har den høyeste regresjonslinja, vil få en lavere designlinje pga. en større varians i datasettet. Materialet til Linje 1 kommer derfor bedre ut totalt.



Figur 19: Eksempel på to sammenfallende linjer med ulik spredning.

5.5.2. Forenklede tester for stigning og skjæringspunkt

Test for felles stigning (parallelle linjer):
Man tester hypotesen H_0: $\beta_1 = \beta_2$, mot H_1: $\beta_1 \neq \beta_2$, og man kan forkaste H_0 hvis:

$$|T| = \frac{|\hat{\beta}_1 - \hat{\beta}_2|}{\hat{s} \cdot \sqrt{\frac{1}{\sum_{i=1}^{n_1}(x_{1i} - \bar{x}_1)^2} + \frac{1}{\sum_{i=1}^{n_2}(x_{2i} - \bar{x}_2)^2}}} > t_{\frac{p}{2}, n_1+n_2-4} \tag{37}$$

Denne testen kan også gjøres ensidig for å se om den ene stigningen er større enn den andre. Da vil man bruke en t-verdi med p, fremfor p/2. Man antar felles varians.

Test for felles skjæringspunkt:
Man tester hypotesen H_0: $\alpha_1 = \alpha_2$, mot H_1: $\alpha_1 \neq \alpha_2$, og man kan forkaste H_0 hvis:

$$|T| = \frac{|\hat{\alpha}_1 - \hat{\alpha}_2|}{\hat{s} \cdot \sqrt{\frac{\sum_{i=1}^{n_1} x_{1i}^2}{n_1 \cdot \sum_{i=1}^{n_1}(x_{1i} - \bar{x}_1)^2}} + \sqrt{\frac{\sum_{i=1}^{n_2} x_{2i}^2}{n_2 \cdot \sum_{i=1}^{n_2}(x_{2i} - \bar{x}_2)^2}}} > t_{\frac{p}{2}, n_1+n_2-4} \tag{38}$$

Denne testen kan også gjøres ensidig på samme måte som for stigningen. Man antar felles varians.

5.5.3. Tester av linjene med matriseregning [14]

Ved hjelp av matriser kan man sette opp en full modell som tar for seg begge datasettene og hvor man kan bestemme om man ønsker å regne med individuelle eller felles verdier for α og β. Funksjonen for modellen blir:

$$Y_{gi} = \alpha_g + \beta_g \cdot X_{gi} + e_{gi} \tag{39}$$

hvor g er gruppe (g = 1, 2) og i er prøvenummer (i = 1, 2, 3, ..., n_g).

For en fullstendig modell kan man sette opp matrisene:

$$X_{Full} = \begin{bmatrix} 1 & 0 & X_{11} & 0 \\ 1 & 0 & X_{12} & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & X_{1n_1} & 0 \\ 0 & 1 & 0 & X_{21} \\ 0 & 1 & 0 & X_{22} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & 0 & X_{2n_2} \end{bmatrix} \tag{40}$$

$$\theta_{Full} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \beta_1 \\ \beta_2 \end{bmatrix} \tag{41}$$

Man kan da sette opp en modell for alle de estimerte Y-verdiene sammen med de estimerte parametrene $\hat{\theta}$. X og θ velges ut i fra modellen man ønsker. Flere varianter av X- og θ-matriser kommer senere i kapittelet.

$$\hat{Y} = X \cdot \hat{\theta} \tag{42}$$

Man kan bruke modellen for å regne ut residualene og RSS (residual sum of squares), som er summen av alle de kvadrerte residualene.

$$\hat{Y} = X(X^tX)^{-1}X^tY = HY \tag{43}$$

$$H = X(X^tX)^{-1}X^t \tag{44}$$

$$e = Y - \hat{Y} = (I - H)Y \tag{45}$$

$$I = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \tag{46}$$

$$RSS = e^t e \tag{47}$$

RSS kan så brukes i hypotesetestene og man forkaster H_0 hvis:

$$F = \frac{(RSS_{H0} - RSS_{H1})/j}{RSS_{H1}/(n_1 + n_2 - 4)} = \frac{(RSS_{H0} - RSS_{H1})}{RSS_{H1}} \cdot \frac{(n_1 + n_2 - 4)}{j} > f_{p,j,n_1+n_2-4} \tag{48}$$

hvor j er antall parametere som testes. Her tilsvarer $RSS_{H1}/(n_1+n_2-4)$ variansen, hvor H_1 er hypotesen for en full modell. Denne vil ha minst varians da den gir en best forklaring av modellen.

F-testen går ut på å teste om endring i forklaringskraft med H_0 mot H_1 er innenfor akseptabelt område slik at man kan si at modellene er like gode.

Test for felles stigning (parallelle):
Man tester hypotesen H_0: $\beta_1 = \beta_2$, mot H_1: $\beta_1 \neq \beta_2$ ved å regne ut RSS verdiene for de ulike scenarioene. Her brukes X_{Full} i formel (47) for RSS_{H1}, mens man for H_0 bruker matrisen X_β:

$$X_\beta = \begin{bmatrix} 1 & 0 & X_{11} \\ 1 & 0 & X_{12} \\ \vdots & \vdots & \vdots \\ 1 & 0 & X_{1n_1} \\ 0 & 1 & X_{21} \\ 0 & 1 & X_{22} \\ \vdots & \vdots & \vdots \\ 0 & 1 & X_{2n_2} \end{bmatrix} \tag{49}$$

$$\theta_\beta = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \beta \end{bmatrix} \tag{50}$$

Man bruker så formel (48) for å bestemme om H_0 skal forkastes. Her vil j være 1 siden det kun er én parameter som testes.

Test for felles skjæringspunkt:
Man tester hypotesen H_0: $\alpha_1 = \alpha_2$, mot H_1: $\alpha_1 \neq \alpha_2$ ved å regne ut RSS verdiene for de ulike scenarioene. Man bruker fortsatt X_{Full} i formel (47) for RSS_{H1}, mens man for H_0 bruker matrisen X_α:

Man bruker så formel (48) for å bestemme om H_0 skal forkastes, med j som 1 siden det fortsatt kun er én parameter som testes.

Test for sammenfallende linje:
For at linjene skal være sammenfallende må de ha både lik stigning og likt skjæringspunkt. Dette tester man i realiteten ikke for ved å teste stigning og skjæringspunkt hver for seg. Selv om det er lite sannsynlig, så kan man risikere at to linjer ikke sammenfaller selv om testene hver for seg går gjennom. En bedre fremgangsmåte er å teste hypotesen H_0: $\alpha_1 = \alpha_2$ og $\beta_1 = \beta_2$ mot H_1: $\alpha_1 \neq \alpha_2$ og $\beta_1 \neq \beta_2$. Dette er det samme som å si at modellen kan reduseres til en generell form $Y=\alpha + \beta X$ og betyr i realiteten at man

kan legge alle verdiene i samme datasett og regne med de vanlige metodene hvis variansen er lik.

Testen utføres ved å regne ut RSS verdiene for de ulike scenarioene. H_1 tilsvarer fortsatt den fullstendige modellen X_{Full}, mens man for H_0 bruker matrisene:

$$X_{\alpha\&\beta} = \begin{bmatrix} 1 & X_{11} \\ 1 & X_{12} \\ \vdots & \vdots \\ 1 & X_{1n_1} \\ 1 & X_{21} \\ 1 & X_{22} \\ \vdots & \vdots \\ 1 & X_{2n_2} \end{bmatrix} \tag{51}$$

$$\theta_{\alpha\&\beta} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{52}$$

Man bruker så formel (48) for å bestemme om H_0 skal forkastes, med j som 2 siden det nå testes for to parametere.

6. Utvikling av programmet

Som nevnt i 2.3 er et design med faner sett på som det mest hensiktsmessige.

Med utgangspunkt i arbeidsflyten i Figur 1 og med tanke på en fanebasert løsning, ble programmet i utgangspunktet delt opp i tre deler:

- Legge inn data.
- Vurdere resultatene grafisk.
- Vurdere resultatene som tekst.

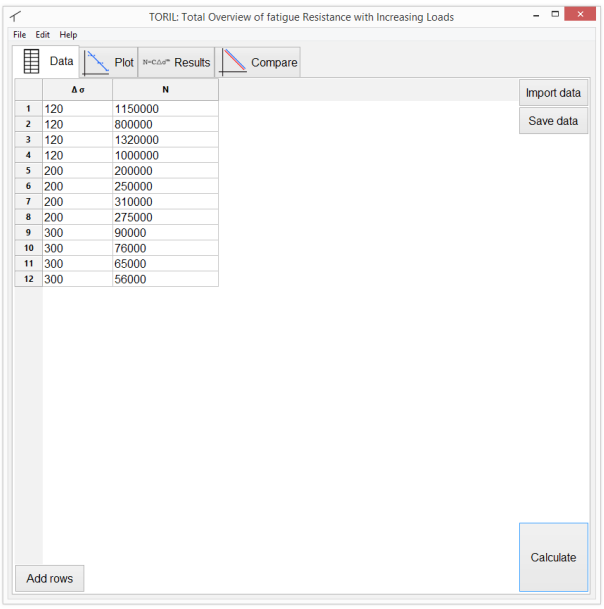
I ettertid ble det også lagt inn mulighet for å sammenlikne linjene fra to datasett.

Utregning er en funksjon og trenger ikke egen fane, noe heller ikke rapporteringsfunksjonen gjør. Rapporteringsfunksjonen får derimot en tilhørighet til den siste tekstbaserte fanen, siden normal arbeidsflyt gjør at den fanen er fanen hvor man er klar for rapportering. Rapportering og utregning kan også skje fra meny eller hurtigtaster.

For bedre synlighet og brukervennlighet ble fanene satt opp med både grafikk og tekst.

6.1. Funksjonalitet

Nedenfor følger en oversikt over de ulike delene av programmet og hva slags funksjonalitet de tilbyr. For en introduksjon til bruk, se "Vedlegg 3: Brukerveiledning".



Figur 20: Datafanen.

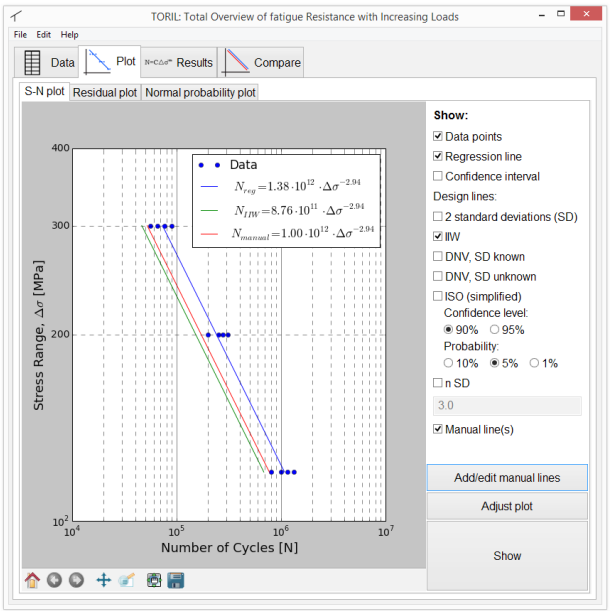
6.1.1. Datafanen

Hovedelementet i datafanen er tabellen man kan legge inn dataen i, med ett utseende som de fleste vil kjenne igjen fra program som Excel. Her kan brukeren legge inn data manuelt eller laste inn data fra fil. Sistnevnte metode krever at dataen er lagret i ett format programmet kan kjenne igjen, enten .txt fil med tab deling ( ) eller .csv fil med semikolon som deling (;). Har man først lagt inn dataen manuelt kan man lagre den til ett gyldig filformat for senere bruk. Dataen som lastes inn legges inn i tabellen og kan endres manuelt ved behov.

Laster man inn data fra fil vil programmet kjøre beregning av S-N linjen automatisk, mens man ved manuell input eller endring må trykke på knappen

"Calculate" (eller en av de andre knappene som viser/oppdaterer informasjonen).

Det er i utgangspunktet satt av plass til 12 datapar, men brukeren kan enkelt øke dette ved å trykke på "Add rows" under tabellen.

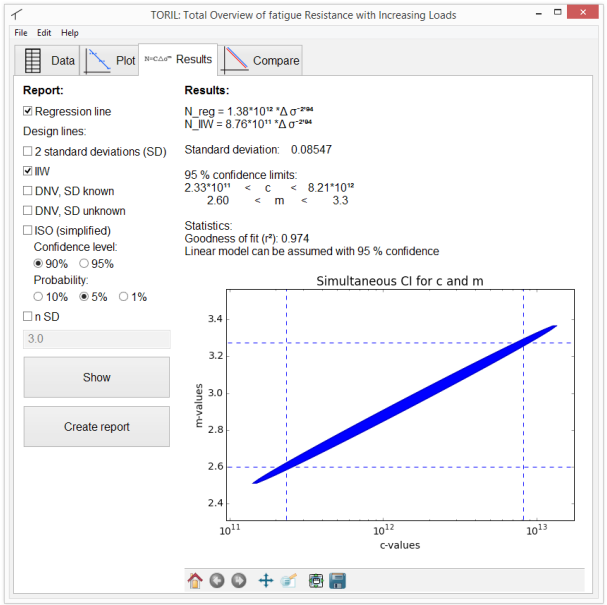


Figur 21: Figurfanen.

6.1.2. Figurfanen

I starten bestod denne fanen kun av fremvisning av S-N kurven med tilhørende valg, men da ISO spesifiserer behovet for residualplott og normal sannsynlighetsplott ble denne fanen utvidet til å inneholde underfaner. Flere plott i en fane ville gjort plottene små og uoversiktlige, så en utvidelse av fanebruken ble sett på som mest hensiktsmessig. Toppen av fanene er kun tekstbaserte og gjør ikke like mye ut av seg, med S-N kurven som den fanen som automatisk dukker opp. På den måten vil nye brukere slippe å forholde seg til de andre plottene, og kan heller velge å se på de etter hvert.

Fanen med S-N kurven er laget for å være mest mulig fleksibel og gi god brukerkontroll. Brukeren har full kontroll over hva som vises i plottet og det er også mulig å legge inn egne linjer basert på en formel, eller designlinjer med fritt valgte antall standardavvik. På denne måten kan programmet også brukes til å lage figurer for bruk i f.eks. undervisningssammenheng eller til illustrasjoner hvor man ikke har ett fullstendig datasett. Brukeren kan også velge utsnitt på figuren, bakgrunnlinjer med mer i en egen meny ("Adjust plot"). Det er også opp til brukeren å velge standard for designlinjen, og brukeren kan velge så mange standarder som ønsket og sammenlikne linjene mot hverandre.

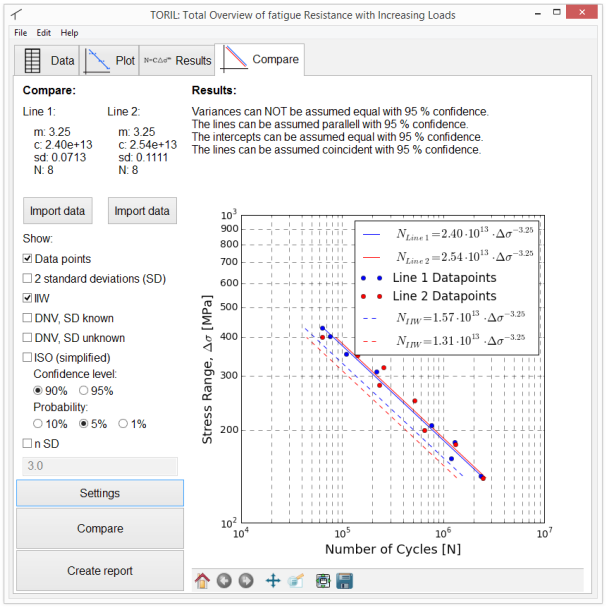


Figur 22: Resultatfanen.

6.1.3. Resultatfanen

Resultatfanen er for tekstbaserte resultater som formelen for linjen, konfidensintervall og statistisk gyldighet. Man kan også her velge hvilke linjer man ønsker å se, og valgene oppdateres mellom fanene. De linjene som er valgt når man velger å rapportere vil bli med i rapporten.

Da verdiene for "c" og "m" er sterkt avhengig av hverandre, og brukeren ikke kan velge disse fritt innenfor konfidensintervallene, ble også en figur med ett simultant konfidensintervall inkludert.

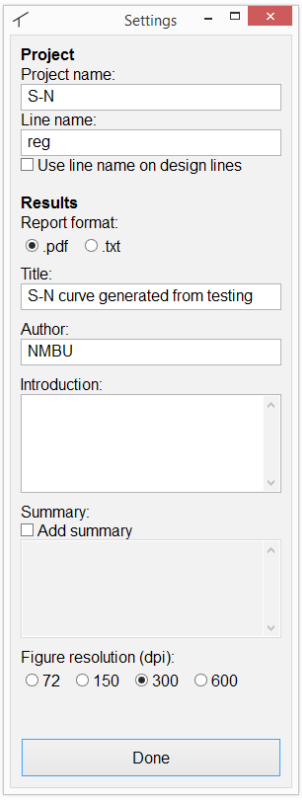


Figur 23: Sammenlikningsfanen.

6.1.4. Sammenlikningsfanen

Denne fanen er beregnet som ekstra funksjonalitet til programmet og er ikke en del av vanlig kjøring. Her trenger man to fullstendige datasett. Hvis man bruker ett datasett i resten av programmet vil den være forhåndslastet som linje 1, men begge linjene kan lastes inn fra filer med testdata. Her stilles samme krav til format som for datafanen, og det går fint an å hente ut dataen fra lagrede innstillingsfiler fra tidligere genererte rapporter. Ellers er denne fanen fristilt fra resten av programmet og samkjører ikke ting som valgte designlinjer. Resultatene herfra skrives også til egen rapport og har egne innstillinger.

Statistisk bakgrunn for å sammenlikne linjer tar for seg om varians, stigning og skjæringspunkt kan antas like og testes hver for seg, og det testes om linjene kan antas sammenfallende ved å teste stigning og krysning sammen. Linjene plottes også sammen for å visuelt kunne vurdere likhet. Siden variansen har så mye å si for designlinjene er det også mulig å sammenlikne designlinjer, samt å vise datapunktene linjen baserer seg på. To linjer kan ha identisk stigning og krysningspunkt, men med ulik varians vil designlinjene skille seg betraktelig fra hverandre.



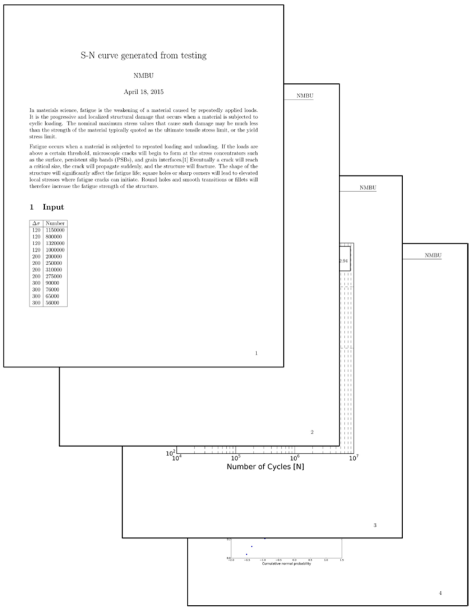
Figur 24: Vindu for innstillinger.

6.1.5. Innstillinger og menyer

Programmet har en vanlig filmeny (”File”, ”Edit” og ”Help”) i toppen av vinduet, som brukeren vil kjenne igjen fra vanlige programmer. I filmenyen kan man lagre og laste inn prosjekter, kjøre/oppdatere resultatene, nullstille programmet og eksportere rapport. Under ”Edit” kan man velge generelle innstillinger for programmet og ”Help” har instruksjoner om både bruk av programmet og valg av designlinjer, samt informasjon om selve programmet. Alle disse valgene har egne tastatursvarveier for de som måtte ønske å bruke dette.

I de generelle innstillingene kan man velge prosjektnavn. Dette danner grunnlaget for navngivning av filer og mapper når man lager rapport. Man kan også velge navngivning av S-N linjen og legge dette navnet til de ulike designlinjene. Her finner man også mange valg til rapporteringen, både filformat og tittel, i tillegg til at man kan angi forfatter og skrive introduksjon og sammendrag. Valg av bildeoppløsning gjelder for figurene som lagres ved rapportering. Direkte lagring av figur med figurens menylinje vil gi lav oppløsning (72 dpi) og den størrelsen plottet er i for øyeblikket.

De spesifikke innstillingene for sammenlikningsfanen har mange like elementer som de generelle innstillingene, da fanen ikke bruker disse. Her finner man også enkle innstillinger for aksene til plottet.



Figur 25: Eksempelrapport

6.1.6. S-N rapportering

Det er ønskelig at rapporten skal være mest mulig til nytte for brukeren og kunne brukes direkte som ett vedlegg. Rapporteringa bruker derfor MikTeX, et program som kan generere PDF fra kode. Det er mulig å endre overskrift og forfatter på rapporten (det ligger inne standardverdier som gjør at brukeren kan lage rapport uten å bry seg om dette), og brukeren kan legge til introduksjonstekst og sammendrag til filen. PDFen vil også inkludere alle figurene fra programmet, samt alle resultater.

Det er også lagt inn mulighet for å velge txt som rapportformat. Rapporten vil da kun inneholde teksbaserte resultater. Dette er lagt inn som en backupløsning hvis brukeren får problemer med MikTeX, ikke ønsker MikTeX for å spare harddiskplass eller av andre grunner foretrekker en tekstfil.

Ved rapportering vil programmet opprette en mappe hvor alle figurene lagres sammen med en innstillingsfil for programmet, samt rapporten. Ved PDF rapport vil det også lagres filer for generering av PDFen. Hvis brukeren er kjent med Tex, er det mulig å gå inn i .tex filen og gjøre egne endringer til rapporten og generere PDF filen på nytt.

Siden programmet oppretter en mappe med flere filer ber eksporteringsfunksjonen kun om en plassering på PC-en og programmet oppretter så en egen mappe med resultatene på den plasseringen. Navnet til mappen bestemmes av prosjektnavnet og de nummereres automatisk hvis det eksporteres til samme sted med samme prosjektnavn.

6.1.7. Rapportering ved sammenlikning

Denne rapporten likner i stor grad rapporten for selve S-N linjen, men har litt mindre informasjon og kun ett plot. Man har også her mulighet for å velge mellom PDF eller txt og det opprettes en egen rapportmappe. Det vil ikke lagres prosjektinnstillinger, men dataen for linjene lagres i hver sin fil.

6.1.8. Lagre prosjektet

Det er mulig å lagre prosjektet. Programmet vil da lagre dataen som er lagt inn samt alle innstillinger og valg som er gjort. Dette inkluderer valg av linjer for visning og innstilling av grenser for plott. Dette lagres automatisk ved rapportering slik at man enkelt kan fortsette fra rapporteringspunktet og gjøre endringer eller hente inn dataen om ønskelig. Sammenlikningsfanen har ikke denne muligheten for lagring av prosjekt da selve linjeverdiene hentes inn fra andre filer og behovet derfor er mindre.

6.1.9. Brukervennlighet

I hele programmet har målet vært å gjøre det enklest mulig for brukeren og prøve å få ting så selvforklarende som mulig. Det er forventet at brukerne ikke vil lese brukermanualen, selv om det legges ved utfyllende manual på norsk og er laget hjelpefiler på engelsk som tar for seg både bruk og valg av designlinjer i selve programmet under "Help" menyen.

Der brukeren selv skriver inn verdier, spesielt tallverdier, er det lagt inn en rekke tester og helgarderinger. Python bruker punktum til desimaltall (komma betyr vektor), men dette korrigeres på programnivå slik at programmet godtar begge deler. Videre kan brukeren velge å skrive store tall fullt ut, eller på eksponentform (f.eks. 1e7). Klarer ikke programmet å kjenne igjen verdien vil brukeren få en feilmelding og den vil ignorere verdien hvis den er i datasettet. Programmet sier også i fra hvis brukeren prøver å legge inn data utenfor gyldighetsområdet til formlene som er brukt.

Da skalaen til plottet er logaritmisk vil programmet ikke klare å gi en nedre aksegrense på 0. Det kan ikke antas at brukeren er kjent med denne begrensningen til log-skalaen, så programmet vil derfor overskrive eventuelle 0-verdier og bruke 1 (10^0) som grense i stede for. Det er antatt at dette vil være nærme brukerens ønske (mindre verdier vil være langt utenfor område for testdataen og gi en dårlig fremstilling av resultatene) og det vil derfor ikke varsles om endringen. Ønsker brukeren noe annet kan det bare endres ved å oppgi en gyldig grense.

Hvis brukeren klarer å trigge en feil som det ikke er gardert for, vil programmet som regel bare ikke respondere. Feilmelding og tilbakemeldinger fra programmet skrives til filene *my_stderr.log* og *my_stdout.log* som det vil være mulig å kontrollere senere.

Hvis brukeren ønsker å starte å legge inn data og innstillinger på nytt er det mulig å nullstille programmet fra filmenyen. Snarveien for dette ble valgt som Ctrl+Shift+C, ikke Ctrl+C, da sistnevnte er en vanlig komando for kopiering av verdier.

Når brukeren prøver å avslutte programmet dukker det opp ett vindu som ber brukeren bekrefte at man virkelig ønsker å avslutte for å hindre at data går tapt ved feilaktig lukking av programmet.

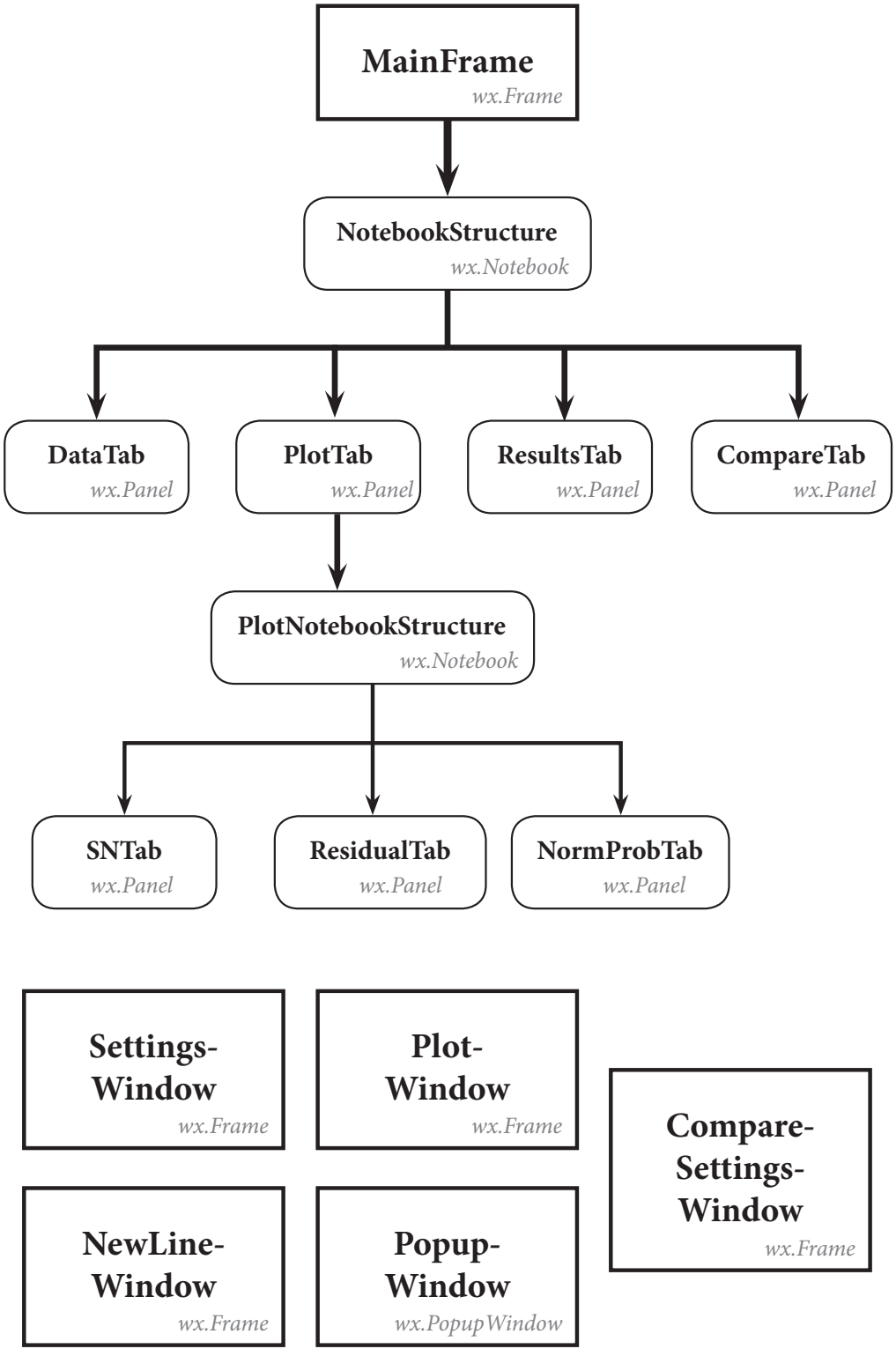
6.2. Brukertesting

Mye av grunnlaget for funksjonaliteten til programmet har blitt utarbeidet i samarbeid med veileder. For å kontrollere att programmet er forståelig i bruk har det også blitt utført en enkel brukertesting. Dette har foregått ved at medstudenter og ansatte ved instituttet har (i) fått en demonstrasjon av programmet med funksjonaliteten det tilbyr, med spørsmål om forbedringspotensiale og uklarheter eller (ii) blitt bedt om å ta i bruk programmet og gjøre enkle endringer, med tilbakemelding på brukeropplevelsen. Metoden som har blitt brukt til testing har variert litt ettersom hvor god tid den som skulle teste programmet har hatt.

Responsen har vært god og det har kun vært behov for mindre justeringer av programmet. En justering var å få programmet til å godta 0 som minimumsgrense på plottet. En annen justering var å legge inn muligheter for tastesnarveier.

6.3. Programmering

Programmet er bygget opp med ett hovedvindu som inneholder fanestrukturen med hovedfanene og man kan se avhengigheten til klassene i figuren under. Hvert kvadrat tilsvarer ett frittstående vindu, og brukergrensesnittet bruker wxPython sine klasser som grunnlag.



Figur 26: Programstruktur med tilhørende klasser.

For at informasjon skal være tilgjengelig mellom fanene er mye av verdiene lagret under MainFrame og NotebookStructure klassene, da alle fanene har tilgang til disse direkte. Ellers er programmet i stor grad satt opp slik at verdier hentes og settes med egne funksjoner (set/get), slik at man kan endre strukturen på programmet uten at man mister referansen til verdier. En klasse spør derfor bare foreldreklassen om en verdi, uten å trenge å vite hvor verdien hører til, så er det opp til foreldreklassen å enten gi verdien direkte eller referere videre oppover til egen forelder.

For å slippe å kjøre alle utregningene på nytt hver gang brukeren ønsker å endre informasjonen som vises, er det lagt inn kode som registrerer når det skjer en endring i f.eks. datafanen og lagrer behovet for å kjøre utregninger på nytt. Programmet sjekker om ett slikt behov har dukket opp hver gang den oppdatterer informasjonen og regner kun ut verdier hvis behovet har dukket opp.

Programmet er også satt opp med ”utf-8” koding, dvs. programmet godtar ”æøå” og andre spesialbokstaver. Dette er spesielt viktig for rapporteringsfunksjonen da forfatteren kan ha disse særbokstavene i navnet sitt. Man forventer engelsk introduksjon og sammendrag, da resten av rapporten bruker engelske formuleringer, men det er også lagt inn mulighet for særbokstaver i disse feltene.

Programmet bruker 95 prosent sikkerhet i sine beregninger av konfidensintervall med mer da dette er en den vanligste verdien for dette. Det vil være mulig å legge inn verdien som brukervalg senere hvis man savner valgmuligheten.

6.3.1. Eksterne klasser

Selve utregningene er trukket ut i en egen fil, models.py, og består av klassene Line, ManualLine og CompareLines.

Line tar for seg alt som har med S-N kurven å gjøre, både utregning av linjen, statistikken og utregning av verdier for plotting. Her er også alle tabellverdiene fra ISO og DNV ført inn og klassen har en rekke metoder for å returnere enkeltverdier eller en samling av verdier, både tall og ferdig tekst til print.

ManualLine er en forenklet versjon av Line og er ment for de manuelle linjene som legges til i plottet. Disse opprettes med linjeverdiene, ikke testdata, og har metoder for å returnere ulike måter å fremstille linjene på, samt verdier for plotting.

CompareLines tar inn to instanser av Line klassen og bruker disse for å kjøre sammenlikninger. Klassen har funksjoner for de ulike statistiske testene, samt en funksjon som returnerer en tekst med alle resultatene.

Malene for rapportering er også trukket ut i en egen fil, report_generator.py, og har klassene ReportGenerator, CompareReportGenerator, ReportGeneratorTXT og CompareReportGeneratorTXT. De to førstnevnte er basert på Tex formatering, har en funksjon for å konvertere .tex fil til PDF og er generelt litt mer omfattende.

Alle malene er basert på bruk av Pythons ”print formatting” med noe som kalles ” placeholders”. Det gjør at malen kan bli satt opp uten spesifikke verdier, kun nøkkelord og type, og få verdiene innsatt senere. Funksjonen ”make_template” forbereder denne, siden ting som antall linjer og datapar kan variere. Verdiene settes inn senere før rapporten genereres.

Det er også en setup.py fil som hører med programmet. Denne brukes for å lage en exe fil av programmet og tar for seg alt som skal inkluderes i programpakken, med unntak av MikTeX som legges til manuelt.

6.4. Navn og logo

Etter forslag fra masterveileder, ble ”Toril” brukt som arbeidsnavn under utviklingen av programmet. Slike dataprogram får ofte vanlige navn (Sintef har utviklet ”Edna”), noe som er lettere å bruke i dagligtale enn bokstavforkortelser. Da ”Toril” også kan brukes som en forkortelse for ”Total Overview of fatigue Resistance with Increasing Loads”, ble navnet værende.

Det ble derfor utviklet en logo med denne forkortelsen, hvor fokus lå på at deler enkelt skulle kunne brukes som gjenkjennbart element i selve dataprogrammet.

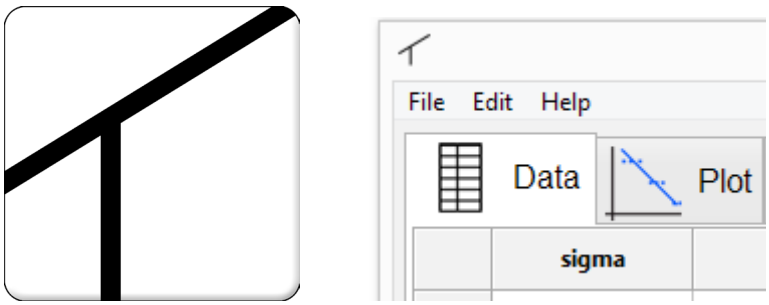
Figur 27:
Utforming av logo.

Etter arbeid med å gjøre T-en spesiell nok til å skille seg ut, men samtidig beholde assosiasjoner til bokstaven, ble resultatet en veldig enkel logo basert på det geometriske forholdet ”det gyldne snitt”. Logoen kan gi assosiasjoner til noe strukturelt, f.eks. bærebjelker eller en kran, samtidig som stigningen til toppen kan minne om en graf. Den går riktignok feil vei, men en realistisk S-N kurve ble forkastet da den fort ville blitt uoversiktlig i så små størrelser som brukes i programmet.

T-en ble kombinert med vanlig tekst for resten av logoen, med en font (skrifttype) som harmonerte med T-en.

Figur 28: Logo med fullstendig betydning.

Da den svarte fargen kan være en utfordring på mørke bakgrunner ble det lagt en hvit bakgrunn på logoen for bruk til selve exe filen. Da vil bokstaven synes uansett bakgrunn.



Figur 29: Eksempler på bruk av logoelementet som ikon og innad i programmet.

7. Verifisering av programmet

ASTM [5] oppgir flere eksempler i standarden sin. Disse eksemplene er med tøyningsverdier, fremfor spenningsverdier, men bruker samme formelverk. Da de oppgir verdier på mellomregninger i programmet er noen av verdiene hentet med ett eget skript som kaller koden for linjen direkte.

Ved hjelp av disse resultatene er verdiene for α og β med konfidensintervall verifisert, samt verdien for standardavvik. Testen for linearitet gir også riktig resultat etter mindre endringer i dataen. Da ASTM ignorerer små forskjeller i tøyningsverdiene får de ett færre antall spenningsnivåer enn programmet regner med. Ved å endre dataen slik at disse verdiene ligger på samme nivå får man en tilnærmet lik F-verdi.

Resultatene har også blitt sammenliknet med resultatene fra Sintef-programmet Edna. Med disse resultatene er både S-N linjen og designlinjen verifisert, samt verdiene for standardavvik og r^2. Da de bruker sentrerte verdier for konfidensintervallet til α og β, som nevnt i slutten av kapittel 5.2, er ikke disse verdiene direkte sammenliknbare.

Resultatene er også kontrollert mot ett eksempel oppgitt fra ISO. Da dette eksempelet bruker spenningsvidde i prosent fremstår ikke plottene optimalt i programmet, men det er kontrollert at standardavvik og r^2 stemmer og at plottene, spesielt residualplottene, stemmer. Normal sannsynlighetsplottet stemmer bra, men residualplottet er litt annerledes. Residualplottet stemmer godt i form, men man ser at det er speilvendt om den horisontale aksen, og akseverdiene er ulike. Det førstnevnte kan tyde på at de har byttet om leddene i beregning av residualene. Studerer man selve linjeplottet ser man at de første residualene er på positiv side av linjen, ikke negativ som ISO sitt plot antyder. Når det gjelder akseverdiene ligger ISO sitt plott langt under hva man forventer av verdier for de standardiserte residualene (se kapittel 5.3), og verdiene samsvarer ikke med de sorterte residualene i normal sannsynlighetsplottet. En alternativ vekting av residualene, studentiserte residualer, gir heller ikke verdier tilsvarende ISO sitt plot. Da dataprogrammet og ISO kommer frem til samme verdi for standardavvik antas det at ISO har brukt en annen verdi for å vekte residualene og at dataprogrammets fremstilling av det standardiserte residualplottet er riktig.

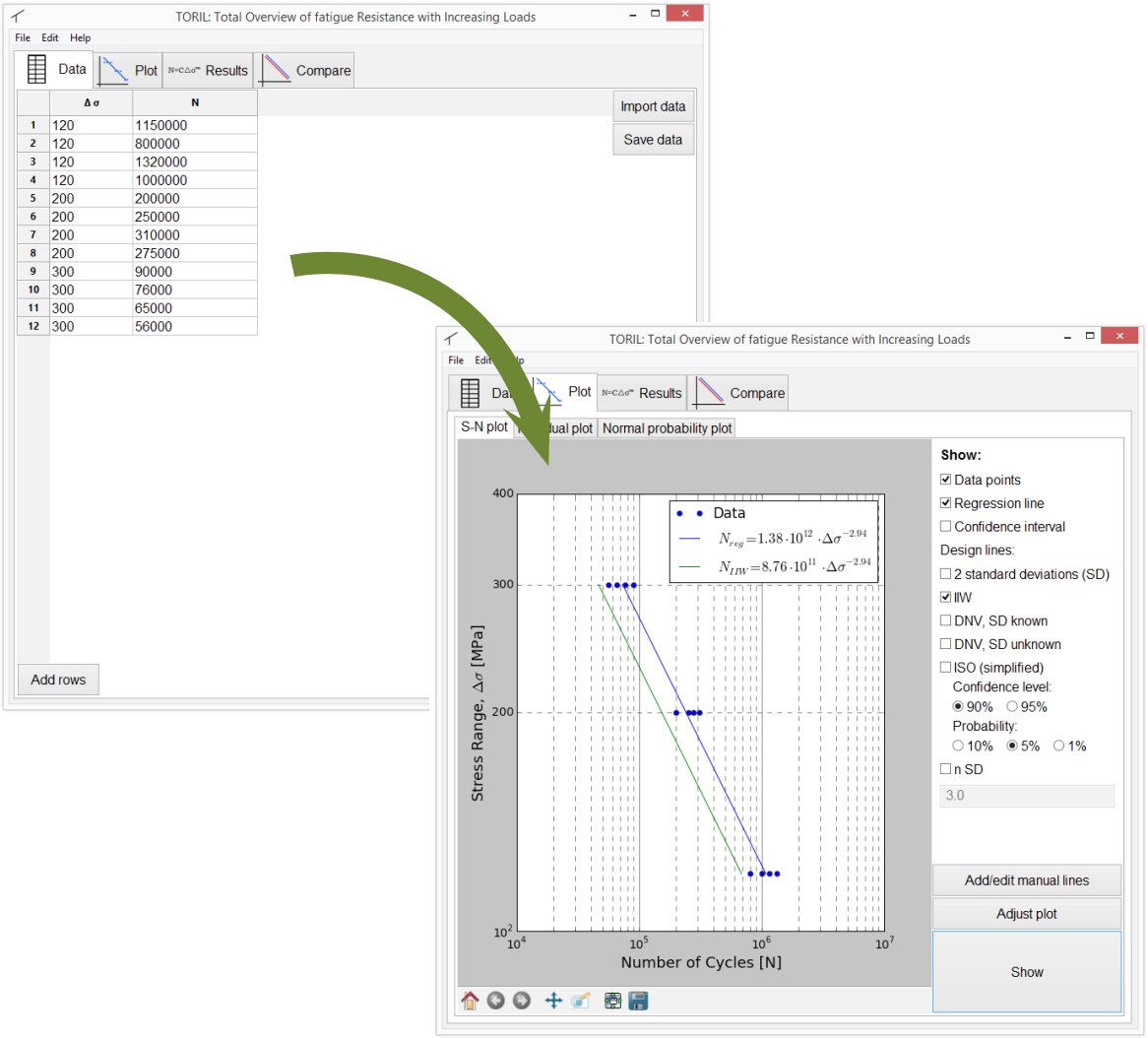
Formlene for sammenlikning av linjene er det ikke funnet noe verifiseringseksempler på, men konklusjonene fra de forenklede testene er testet mot konklusjonen til matrisetestene og stemmer godt over ens. Det er også blitt utført flere visuelle tester for å kontrollere at konklusjonene stemmer.

Programmet har blitt kontrollert at kjører på en rekke ulike Windows-maskiner, både med Windows 7 og 8, og både 32 og 64 bit. Programmet har også blitt kjørt på en rekke ulike måter for å finne og rette opp eventuelle feil og sørge for at all funksjonaliteten er til stede.

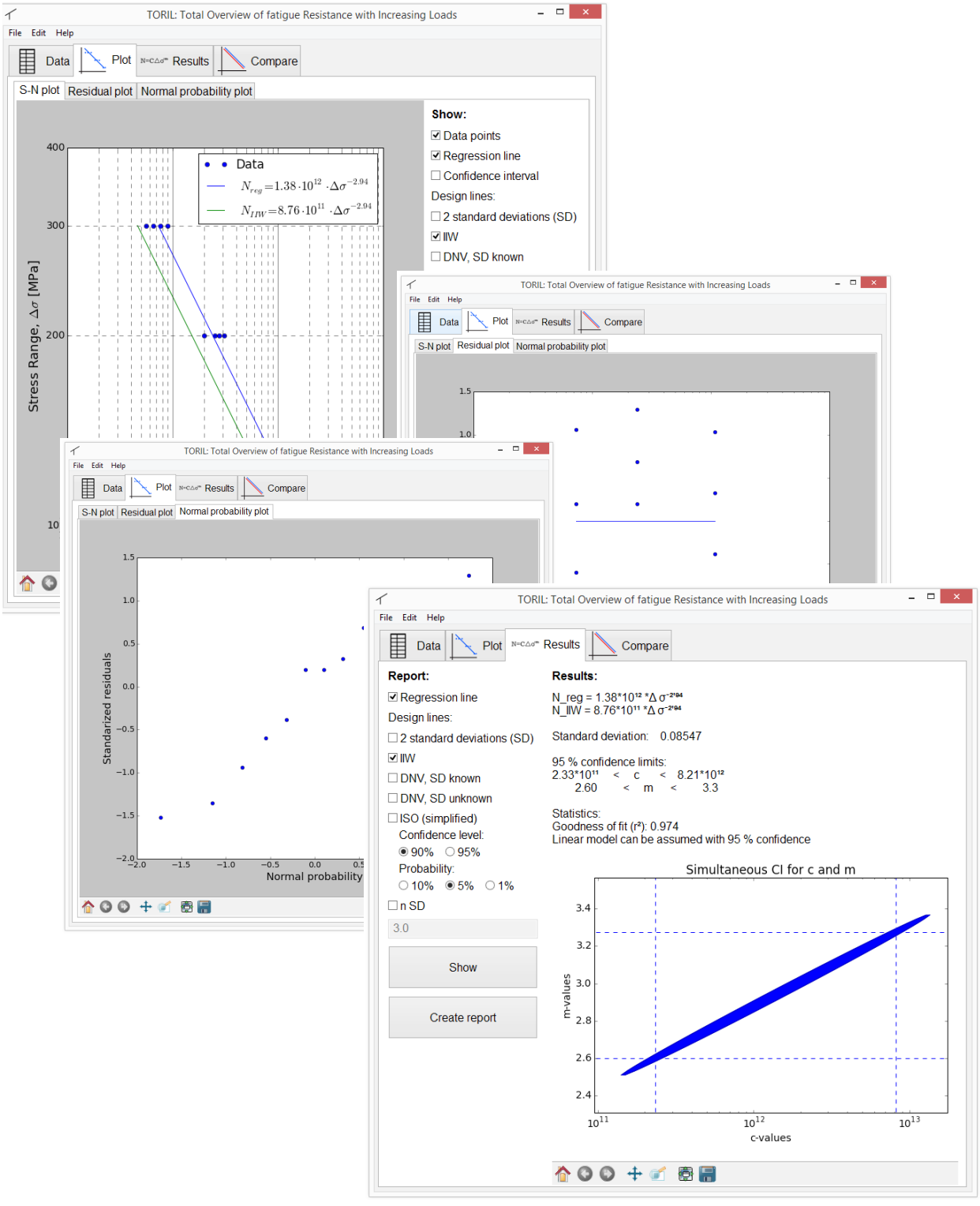
8. Presentasjon av programmet

8.1. Visualisering

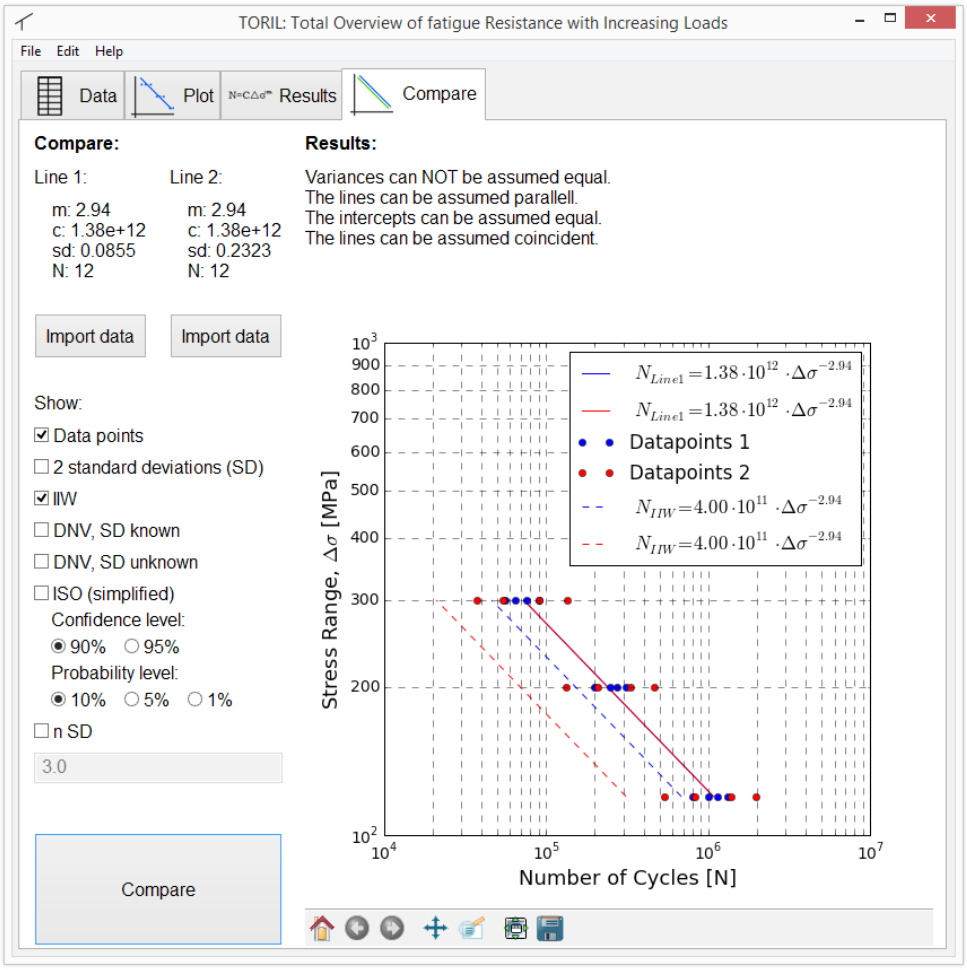
Under følger en visualisering av hvordan programmet er satt opp. For bruk, se "Vedlegg 3: Brukerveiledning".



Figur 30: Konvertering av data til grafisk fremstilling.



Figur 31: Resultater for S-N linjen.



Figur 32: Sammenlikningsfanen.

8.2. Forbedringer

Det finnes flere aspekter ved utmattingsberegning. Flere av standardene tar for seg utmattingstesting for å verifisere at designlinjen man ønsker kan brukes og dette kan være en naturlig utvidelse til programmet

Videre krever programmet ett fullt datasett for å beregne designlinjene siden den trenger standardavvik og antall prøver i denne beregningen. Det kan være aktuelt å se på muligheten for å legge inn linjen som en formel med standardavvik og antall prøver oppgitt som ett alternativ til ett fullt datasett.

Det kan også være interessant å se på utmatting utenfor området som er tatt for seg i denne rapporten, både lavsykel og høysykel, i tillegg til en annengradslikning som forklaring til dataen.

Videre kan det også være aktuelt å legge til beregninger av levetid, f.eks. med Miner Palmgrens modell for kumulativ skade.

Programmet er foreløpig kun tilpasset Windows-brukere. Videre arbeid kan inkludere å lage programmet kjørbart på Mac og Linux-maskiner.

Koden kan også forkortes og optimaliseres. Da det har blitt lagt til ny funksjonalitet underveis har det blitt flere liknende funksjoner som kan kombineres hvis man tar jobben med å rydde koden.

9. Diskusjon

9.1. Gjeldende regelverk og statistisk bakgrunn

Standardene har ganske store forskjeller seg i mellom på omfang og innhold, og kan til tider være krevende å tolke. Britisk Standard og DNV er begge omfattende med en hovedvekt på beregning av utmatting på strukturer ut i fra eksisterende kurver. Spesielt Britisk Standard stiller veldig svakt på beregningsdelen og har hovedfokus på å validere kurver basert på testing fremfor generering av nye. DNV tar heller ikke for seg beregning av selve S-N kurven, men har til gjengjeld en designlinje som virker gjennomtenkt. ASTM står i sterk kontrast til disse med sine 7 sider. Her finner man en lettfattelig og oversiktlig gjennomgang av den statistiske analysen av dataen med ett fullstendig formelverk, tabellverdier og eksempler. ASTM tar derimot ikke for seg designlinjen. ISO har også fokus på å lage S-N kurver fra måledata, men er betydelig mindre lettlest. Den tar for seg testing både med spenning og med sykler som forklaringsvariabel og begrenser seg heller ikke kun til lineære kurver. Man finner til dels formler for S-N kurven og de oppgir også en formel som kan brukes som designlinje.

Selv om flere av standardene ikke tar for seg utregning av S-N kurven er dette noe som kan regnes ut med lineær regresjonsanalyse og er ikke noe som varierer. I beregning av designlinjen finner man større forskjeller. Britisk Standard er den minst konservative, med sine 2 standardavvik uavhengig av antall prøver, mens DNV er den mest konservative hvis man ikke legger til grunn ett kjent standardavvik. Med ett kjent standardavvik blir sikkerhetsmarginen til DNV forholdsvis lik IIW, men med litt variasjoner basert på antall tester. ISO gir ikke noen direkte anbefalinger til sikkerhetsmarginer, men oppgir en formel hvor brukeren velger en sikkerhetsmargin ut i fra pålitelighet til estimeringen og dens konfidensnivå fra tabell. Ettersom hvilke verdier man velger kan man både få den mest og den minst konservative linjen. Ønsker man en linje tilsvarende de andre standardene med tanke på sikkerhet bør derimot linjene med 10 % sannsynlighet unngås, da de havner langt under 2 standardavvik når man kommer opp på ett tosifret antall prøver. I ett eksempel i standarden bruker de en linje med 95 % konfidensnivå og 1 % sannsynlighet, noe som gir en veldig konservativ linje. ISO sin standard kan derfor være den bruksmessig mest krevende da den setter større krav til valg fra brukeren.

Kan man si at standardavviket er kjent, og har 10 eller flere prøver, vil DNV og IIW stille forholdsvis likt og begge kan være gode valg. DNV gir en større sikkerhetsmargin ved få prøver. Britisk Standard sin anbefaling om 2 standardavvik kan fremstå litt for enkel og bør derfor helst kun brukes ved ett høyt antall prøver eller der man ikke trenger like stor sikkerhet mot brudd.

For sammenlikning av kurvene er det brukt formler fra andre rapporter på området og generell statistikklitteratur, da dette ikke er noe som er unikt for utmatting. Ekstern statistikklitteratur har vært

essensielt der man ikke bruker de forenklede formlene, da rapportene kan være vanskelig å tolke, både for de med og de uten statistikkbakgrunn.

9.2. PROGRAMMET

Programmet har vært i stadig utvikling og endring underveis. Det har blitt lagt til mer funksjonalitet underveis, men noe har også blitt fjernet eller forenklet. Det har vært fokus på en balanse mellom brukervennlighet og fleksibilitet.

Det er opp til brukeren å velge hva slags standard eller metode som ønskes for designlinjen, men programmet og brukermanualen inneholder en beskrivelse av de ulike metodene så brukeren kan gjøre ett informert valg. Sikkerhetene er fastsatt hos IIW til den anbefalte 97,7 % sikkerheten og det brukes 95 % sikkerhet i tester og konfidensintervaller. Disse verdiene er så vanlige å bruke at det ble sett på som unødvendig å legge dette valget til brukeren. Ønskes andre sikkerheter kan brukeren regne ut korreksjonsfaktor manuelt og bruke ”n SD” alternativet for designlinjen eller legge den inn som en manuell linje.

Lagring av enkeltfiler følger vanlig system med at brukeren skriver filnavn og lagrer direkte, men rapporteringen kan fremstå litt uvant. Siden det skal lagres en mappe med resultater bes brukeren om en plassering, fremfor filnavn, og programmet oppretter selv en ny mappe med tilhørende filer. Det ble vurdert å bruke samme lagringsmåte med filnavn, som da ville gjelde rapporten, og hvor ekstra filer ble plassert på samme lokasjon. En utfordring med dette er at filene som opprettes fort kan bli oversett hvis rapporten genereres i en mappe med mye innhold og det er ikke like tydelig for brukeren at det genereres ett sett med filer. Navngivning og å unngå at filene overskriver andre filer ville også blitt utfordrende.

Rapporteringsfunksjonen fungerer bra. Med PDF-løsningen får man greske bokstaver og figurene inkludert i rapporten. Det kan til tider være mye luft i rapporten ettersom hvor mye man legger inn av resultater og tekst før og etter. Det er derimot vanskelig å hindre uheldig sidedeling hvis man ikke legger inn sidedeling på faste plasser, så fordelene sees på som større enn ulempene.

Opprinnelig fulgte designlinjene en fast rekkefølge av farger når brukeren valgte ekstra designlinjer. Manuelle linjer ville også følge samme systemet. En utfordring med denne metoden er at det er ett begrenset antall tydelige farger som enkelt kan skille seg fra hverandre. Lyse farger kan lett bli borte og nyanser som ligger for nære hverandre kan bli for like med dårlig skjerm eller printkvalitet. Med ett lite antall linjer vil ikke dette være ett problem, men da brukeren kan velge opp til 6 designlinjer, og ett ubegrenset antall manuelle linjer, kunne det bli en utfordring. Utseende på linjene ble derfor endret til å bruke ulike former for stiplede linjer (4 ulike, fra heltrukket til prikker), som så går over til å få dekorasjoner på endepunktene ved ytterligere antall. Siden linjene nå skilles på form ble alle designlinjene satt grønne, mens det er brukt rød for manuelle linjer. På den måten er det også tydelig hvilke linjer programmet har regnet ut og som dermed følger en standard.

Brukergrensesnittet har ikke en like god implementering av spesialtegn som greske bokstaver og hevet/senket skrift som plottene og PDF-rapporteringen. Greske bokstaver er blitt implementert, og hevet skrift er implementert i den grad det lar seg gjøre i resultatfanen. Senket skrift var ikke mulig.

Det er lagt inn fri navngiving av selve regresjonslinjen, men da programmet inneholder så mange standarder blir standardnavnet beholdt i plottinga. Det er mulig å legge navnet til regresjonslinja foran standardnavnet, men ikke ha navnet alene. Det vil ellers ikke være mulig å skille linjene. Ønsker en fri navngivning kan man heller legge inn designlinjen som en manuell linje.

Programmet fungerer bra og det sees på som en stor fordel at det kan kjøres direkte fra minnepenn uten noe form for installasjon av selve programmet eller andre programmer som kjøringen avhenger av. Med tanke på beregningstid hadde programmeringsspråket C++ vært raskere, men da beregningene i bakgrunn er av begrenset omfang og datasettene pleier å være små, vil dette ikke være til stor sjenanse for brukeren. Det kan være litt mer merkbart når man kjører programmet fra minnepenn. Det er til gjengjeld betraktelig raskere å programmere i Python, noe som gjør at programmet har betydelig mer funksjonalitet enn det ville hatt om det ble programmert i C++.

Programmet sees på som en stor forbedring til dagens metodikk med oppsett av dataen i Excel hvor brukeren må finne og sette opp formlene manuelt. I Excel vil utseendet til resultatene også i stor grad avhenge av brukerens kunnskap om programmet. Programmet som er utviklet krever liten forståelse for statistikken som ligger i bakgrunn og brukeren kan i stor grad bruke rapportene direkte. Den grafiske kvaliteten er høy og formlene for linjene er formatert pent i PDF rapportene.

Resultatene for S-N linjen stemmer overens med kjente verdier. Det er ikke tilsvarende sammenlikningsmuligheter for sammenlikningen av linjene, men konklusjonene er vurdert mot både visuelle vurderinger og vurderinger med forenklede formler.

Programmet har også utviklingspotensiale og det er mulig å legge til mer funksjonalitet i form av nye faner, eller å utvide funksjonalitet i de eksisterende. Det vil være en fordel å brukerteste programmet i større skala for å få frem eventuelle uklarheter i programmet og se om programmet kan optimaliseres.

10. Konklusjon og anbefalinger

Programmet er velfungerende og rapporten gir en enkel oversikt over de ulike standardene på området og over statistikken som ligger til grunn for S-N kurven. Statistikkdelen er satt opp som en hjelp for å få oversikt over formelverk man kan bruke og hva de ulike standardene anbefaler.

Standardene har store forskjeller innad, med styrker og svakheter. ASTM kan trekkes frem som en veldig ryddig og god oversikt over behandlingen av testdataen med formler for utregning, mens DNV og rapporten fra IIW begge kommer med gode forslag til designlinjen. Britisk Standard har ett veldig begrenset innhold når det kommer til beregning av SN-kurven og valg av designlinjer. ISO fokuserer riktignok på analyse av utmattingsdata, men rapporten er unødvendig komplisert for bruk til lineære sammenhenger og designlinjen kan være vanskelig å forstå.

10.1. Anbefalinger

Programmet fungerer på en tilfredsstillende måte og sees på som en betydelig forbedring til dagens metode med bruk av regneark.

- Det kan anbefales at man bruker programmet til beregning og fremvisning av måledata fremfor å regne ut dette manuelt i programmer som Excel da dette i mange tilfeller gir bedre visuelle resultater og resultatene er verifisert mot kjente resultater.
- Ønsker man å anskaffe en standard for å få en oversikt over beregningsgrunnlaget til S-N kurven anbefales standarden til ASTM.
- Ønsker man å anskaffe en standard for å få anbefalinger til designlinjen anbefales DNV, mens IIW kan nevnes som en relevant rapport.

10.2. Videre arbeid

Ønsker man å arbeide videre med programmet kan følgende elementer anbefales å se mer på:

- Brukertesting i større skala vil være viktig fremover for å få flere synspunkter på programmet som arbeidsverktøy, ikke bare synspunkt på førsteinntrykk.
- Behovet for beregninger utenfor området programmet dekker og behov for en kvadratisk modell bør undersøkes og eventuelt implementeres i programmet.
- Man bør se om det er behov for versjoner av programmet som kan kjøres på Mac eller Linux.
- Programmet kan utvides med en fane for verifisering av designlinje fra testing.
- Programmet kan gjøres mer fleksibelt ved at brukeren kan føre inn S-N linje med antall prøvestaver og standardavvik der man ikke har fullt datasett, men ønsker designlinje.
- Man kan også inkludere andre aspekter som beregning av levetid med Miner Palmgren.

11. Litteraturreferanser

11.1. Skriftlige kilder

1. Burge, S., *The Systems Engineering Tool Box: Pugh Matrix (PM)*, 2009, 15 sider.
2. Bøe, J. K., *Konsept- og Produktrealisering,* NMBU, Ås, 2014, 180 sider.
3. Juvinall, R. C., Marshek, K. M., *Fundamentals of Machine Component Design*, Fourth Edition, 2006, John Wiley & Sons, New Jersey, ISBN: 978-0-471-74285-2, 769 s.
4. Waløen, Å. Ø., Maskindeler, Bind 1, 1989, Tapir forlag, Trondheim, ISBN: 82-519-0884-1, 271 s.
5. ASTM E739-10, *Standard Practice for Statistical Analysis of Linear or Linearized Stress-Life (S-N) and Strain-Life (ε-N) Fatigue Data*, 2010, ASTM International, Pennsylvania
6. DNV-RP-C203:2012, *Fatigue Design of Offshore Steel Structures*, 2012, DNV, Høvik.
7. BS 7608:2014, *Guide to fatigue design and assessment of steel products*, 2014, British Standards Institution, London.
8. ISO 12107, *Metallic materials - Fatigue testing - Statistical planning and analysis of data*, 2012, International Organization of Standardization, Geneva, Sveits.
9. Rausland, M., *Statistical Analysis of Fatigue Test Data*, SINTEF-report STF18-A81047, 1981.
10. Ørjaseter, O., *Planlegging av utmattingsforsøk og statistisk evaluering av resultater*, SINTEF-report STF24 F97296, 1997.
11. Schneider, C. R. A., Maddox, S. J., *Best Practice Guide on Statistical Analysis of Fatigue Data*, IIW-XII-WG1-114-03, 2003, International Institute of Welding, Cambridge, UK.
12. Montgomery, D. C., Peck, E. A. og Vining, G. G., *Introduction to Linear Regression Analysis*, Fourth Edition, 2006, John Wiley & Sons, New Jersey, ISBN: 978-0-471-75495-4, 612 s.
13. Weisberg, S, *Applied Linear Regression*, Second Edition, 1985, John Wiley & Sons, New Jersey, ISBN: 0-471-87957-6, 324 s.
14. Draper, N. R., Smith, H., *Applied Regression Analysis*, Second Edition, 1981, John Wiley & Sons, New Jersey, ISBN: 0-471-02995-5, 709 s.

11.2. Internettkilder

15. Bilde av utmattingsbrudd: http://reliabilityweb.com/ [Nedlastet 30.04.2015]
16. Bilde av stag fra Alexander Kielland: http://upload.wikimedia.org/ [Nedlastet 30.04.2015]

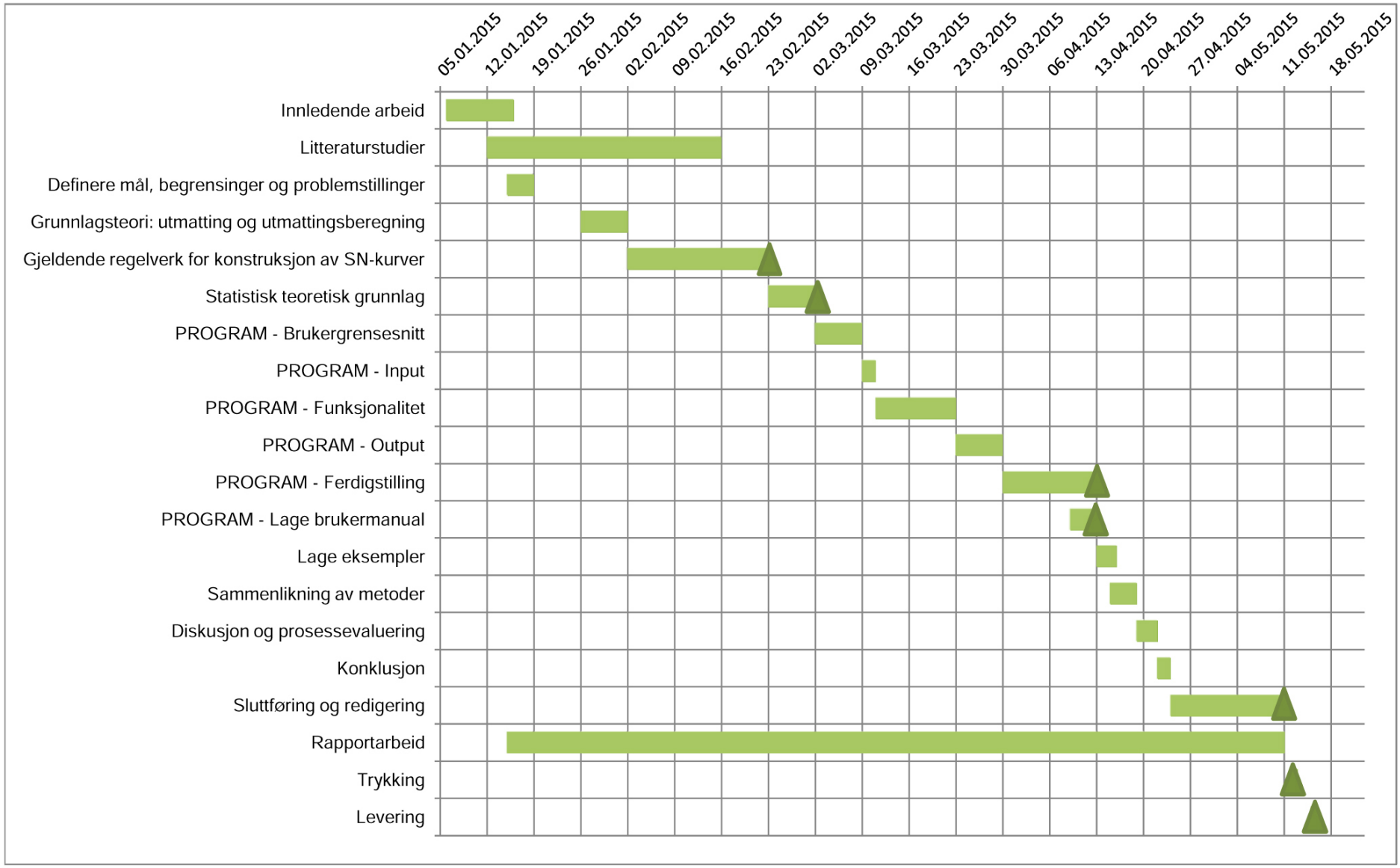
12. Vedlegg

Vedlegg 1: Tids og arbeidsplan med milepæler.
Vedlegg 2: Refleksjonsnotat
Vedlegg 3: Brukerveiledning
Vedlegg 4: Eksempelrapport: S-N kurven
Vedlegg 5: Eksempelrapport: Sammenlikning av kurver
Vedlegg 6: Kildekode

- Toril.py
- models.py
- report_generator.py
- setup.py

Fremdriftsplan

Aktivitet	Startdato	Ant. dager	Sluttdato
Innledende arbeid	06.01.2015	10	15.01.2015
Litteraturstudier	12.01.2015	35	15.02.2015
Definere mål, begrensinger og problemstillinger	15.01.2015	4	18.01.2015
Grunnlagsteori: utmatting og utmattingsberegning	26.01.2015	7	01.02.2015
Gjeldende regelverk for konstruksjon av SN-kurver	02.02.2015	21	22.02.2015
Statistisk teoretisk grunnlag	23.02.2015	7	01.03.2015
PROGRAM - Brukergrensesnitt	02.03.2015	7	08.03.2015
PROGRAM - Input	09.03.2015	2	10.03.2015
PROGRAM - Funksjonalitet	11.03.2015	12	22.03.2015
PROGRAM - Output	23.03.2015	7	29.03.2015
PROGRAM - Ferdigstilling	30.03.2015	14	12.04.2015
PROGRAM - Lage brukermanual	09.04.2015	4	12.04.2015
Lage eksempler	13.04.2015	3	15.04.2015
Sammenlikning av metoder	15.04.2015	4	18.04.2015
Diskusjon og prosessevaluering	19.04.2015	3	21.04.2015
Konklusjon	22.04.2015	2	23.04.2015
Sluttføring og redigering	24.04.2015	17	10.05.2015
Rapportarbeid	15.01.2015	116	10.05.2015
Trykking	12.05.2015	1	12.05.2015
Levering	15.05.2015	1	15.05.2015


05.01.2015
12.01.2015
19.01.2015
26.01.2015
02.02.2015
09.02.2015
16.02.2015
23.02.2015
02.03.2015
09.03.2015
16.03.2015
23.03.2015
30.03.2015
06.04.2015
13.04.2015
20.04.2015
27.04.2015
04.05.2015
11.05.2015
18.05.2015
Innledende arbeid
Litteraturstudier
Definere mål, begrensinger og problemstillinger
Grunnlagsteori: utmatting og utmattingsberegning
Gjeldende regelverk for konstruksjon av SN-kurver
Statistisk teoretisk grunnlag
PROGRAM - Brukergrensesnitt
PROGRAM - Input
PROGRAM - Funksjonalitet
PROGRAM - Output
PROGRAM - Ferdigstilling
PROGRAM - Lage brukermanual
Lage eksempler
Sammenlikning av metoder
Diskusjon og prosessevaluering
Konklusjon
Sluttføring og redigering
Rapportarbeid
Trykking
Levering
Milepel
Varighet av aktivitet

Refleksjonsnotat

Koden har vært under kontinuerlig forbedring og blitt utvidet betydelig underveis. Hadde jeg visst omfanget og all funksjonaliteten da jeg begynte å programmere hadde jeg nok end opp med en kortere og mer oversiktlig kode. Nå er det en del funksjoner som gjør mye av det samme, men ikke likt nok til å kunne korte vekk. Allikevel har jeg slettet mye kode underveis og restrukturert deler av programmet ved flere anledninger.

Standardene har vært krevende. Spesielt ISO er veldig uoversiktlig og selv statistikerne på universitetet har hatt vansker med å skjønne hva standarden sier til tider. Noe av utfordringen er også at det ikke er en felles konvensjon for navngivning og formlene kan gjerne være formulert på ulike måter i ulike rapporter og bøker. Jeg har derfor brukt en del tid på å sette meg inn i bakgrunnen for beregningen av linjen for å være komfortabel med formlene jeg bruker. Dette hadde nok vært lettere om jeg hadde hatt flere fag i statistikk å bygge på.

Jeg har ikke fulgt tidsplanen veldig grundig og har endt opp med å gjøre mye parallelt. Etter at jeg har lest meg opp på ett område har jeg som regel ført det inn i rapporten, og så programmert funksjonene for å kontrollere at resultatene stemmer og at jeg skjønner hvordan den skal brukes. Dette har derimot passet godt i dette prosjektet da jeg har fått kontrollert forståelsen, fått god kontroll på programmet og visst hvor mye jeg kunne utvide.

Fordelen med å gjøre flere aspekter ved oppgaven på en gang er også at det har gitt meg litt variasjon og avbrekk. Standardene har krevd mye energi og det har hjulpet med litt variasjon.

Det har vært veldig lærerikt med ett så omfattende prosjekt og det har vært gøy å kunne gjøre en oppgave så grundig. Skulle jeg startet på nytt med samme prosjekt hadde det jo gått lettere da jeg har en mye større forståelse nå, men hadde jeg startet på noe helt nytt og ukjent er jeg usikker på hvor mye jeg hadde hatt nytte av å endre arbeidsmetoden.

Brukerveiledning

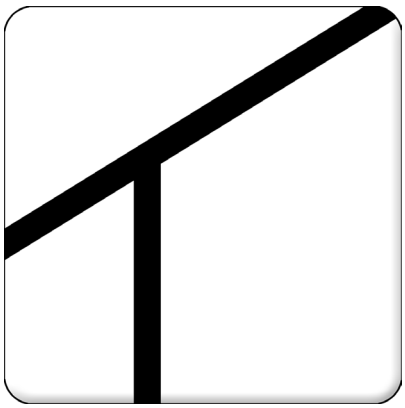
TORIL

Total Overview of fatigue
Resistance with Increasing Loads

NMBU - våren 2015

Innhold

1. Om programmet



Programmet er laget som en del av en masteroppgave, våren 2015. Oppgaven tar for seg statistisk bakgrunn for utarbeidelse av S-N kurver og går gjennom hva de ulike standardene på området setter som krav for designlinjer.

Programmet bruker følgende standarder:

- ASTM-International E739-10
- DNV-RP-C203
- BS 7608:2014
- ISO 12107

I tillegg er formelen for designlinje fra IIW-XII-WG1-114-03 inkludert.

S-N kurven angis på formen: $N = c \cdot \Delta\sigma^{-m}$ og tegnes i ett dobbeltlogaritmisk diagram.

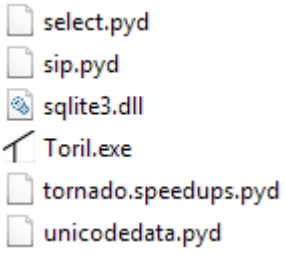
Begrensninger

Programmet er beregnet for resultater i området 10^4 til $5 \cdot 10^6$ sykler og tar ikke for seg knekk i S-N kurven. Se masteroppgaven og de enkelte standardene for ytterligere begrensninger.

Brukeren er selv ansvarlig for at programmet brukes riktig og at resultatene stemmer. Programmet kan fritt brukes av ansatte og studenter ved NMBU, men kan ikke distribueres utenfor campus uten tillatelse.

Bruk

Programmet krever ikke installasjon og kan kjøres direkte fra minnepenn eller legges lokalt på PCen. Det er viktig å beholde mappestrukturen i programmet og la alle filer være uendret.

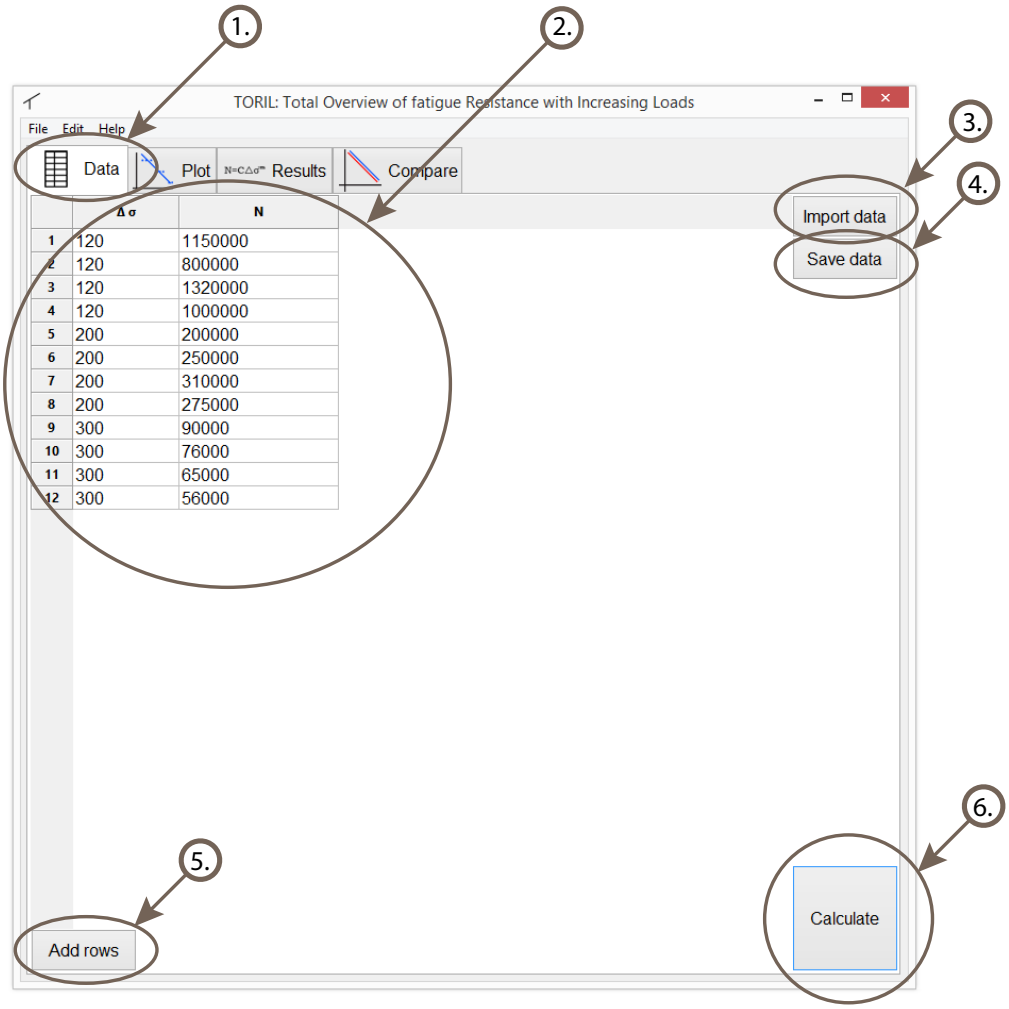


For å kjøre programmet, dobbelklikk på Toril.exe, som finnes i hovedmappen Toril. Ønskes en snarvei til skrivebordet kan man gjøre dette ved å høyreklikke på logoen og velge "Create shortcut" eller lignende. Denne kan fritt flyttes til skrivebordet. Merk at denne kun er gyldig så lenge plasseringen til programmet er det samme.

Hvis programmet allerede er lagt inn på PC-en, men uten snarvei, kan programmet lokaliseres ved å trykke på Windows-ikonet og søke etter Toril.exe. I Windows 8 vil maskinen automatisk starte ett søk når man begynner å skrive og man trenger ikke å lokalisere søkefunksjonen.

2. Legge inn data

Den første fanen (1.) i programmet er der man legger inn data. Dette kan gjøres ved å føre dataen direkte inn i tabellen (2.), eller ved å lasta inn data fra fil(3.). Programmet godtar verdier på både normal form og på eksponentform (f.eks. 1e7).



1. Faneplassering
2. Visning/ redigering av data.
3. Import av data
4. Lagring av data
5. Legge til rader
6. Bearbeide data.

Filformat som støttes er .csv og .txt, hvor dataene er delt med semikolon(;) i førstnevnte og tab ( ) i sistnevnte. Spenning angis først og filen kan starte med en overskrift. Spenningsverdiene kan også hentes ut i fra lagrede prosjektfiler.

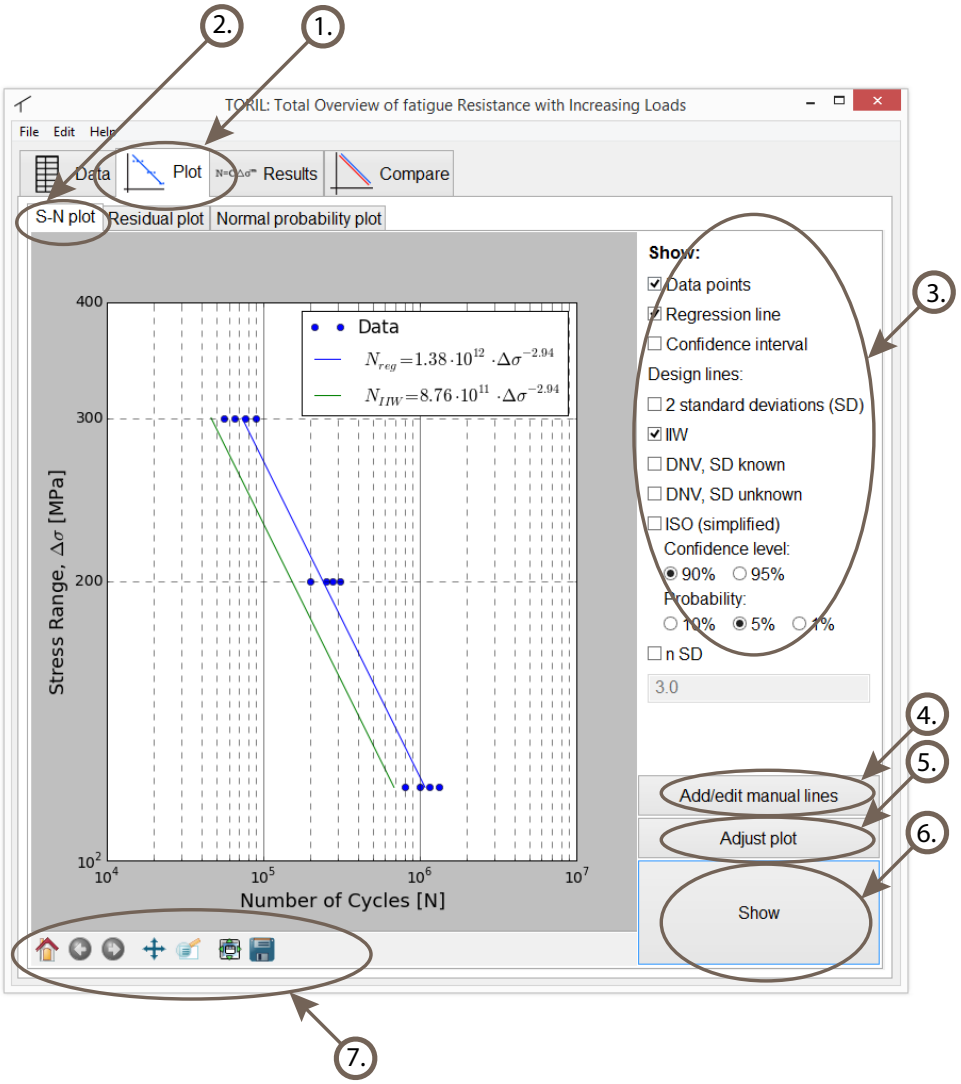
Dataen kan lagres til fil i ønsket filformat (.csv eller .txt) ved å trykke på "Save data" (4.). Skriver man ikke ett filformat til filnavnet brukes .csv som standard.

Trenger man flere rader kan dette legges til med knappen nede til venstre (5.).

Når dataen er klar brukes "Calculate"(6.) for å regne ut verdier og tegne grafer. Resultatene kan evalueres under "Plot" og "Results", eller man kan eksportere en rapport umiddelbart via "File - Create report". Mer om dette i kapittel "5. Eksportere rapport".

3. Evaluere plott

Plottene kan vurderes visuelt under fanen "Plot" (1.), og den første fanen (2.) viser selve måleresultatene med tilhørende regresjonslinje.



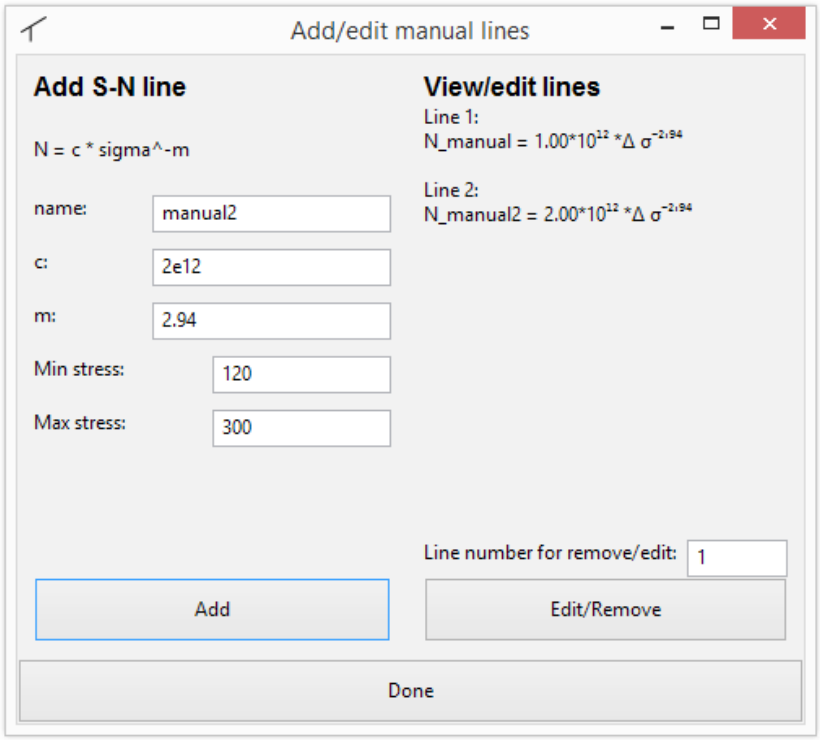
1. Faneplassering
2. Underfane
3. Valg for visning
4. Legge til eller endre egne linjer
5. Justere plottet
6. Vise plot
7. Meny for plot

Man kan velge hva som skal vises i plottet og kan legge til én eller flere designlinjer. Dette gjøres ved å huke av de respektive boksene (3.) og så trykke på knappen "Show" (6.). Den nederste boksen, "n SD", gir mulighet for å legge til ett valgfritt antall standardavvik som sikkerhet.

Videre er det mulig å legge egne linjer til plottet via knappen "Add/edit manual lines" (4.). Disse vil kun være for visualisering/sammenlikning og gir ikke mulighet for tilhørende designlinjer.

Ønskes et annet utsnitt, eller andre innstillinger for plottet, kan dette velges under "Adjust plot" (5.). Det er også mulig å bruke menylinjen i bunn av plottet (7.) for å zoome inn på områder med mer. Hjem eller tilbakeknappen brukes for å gå tilbake til opprinnelig utsnitt og det er mulig å lagre plottet. Utsnitt valgt med denne menyen vil ikke bli brukt i eksport av resultatene.

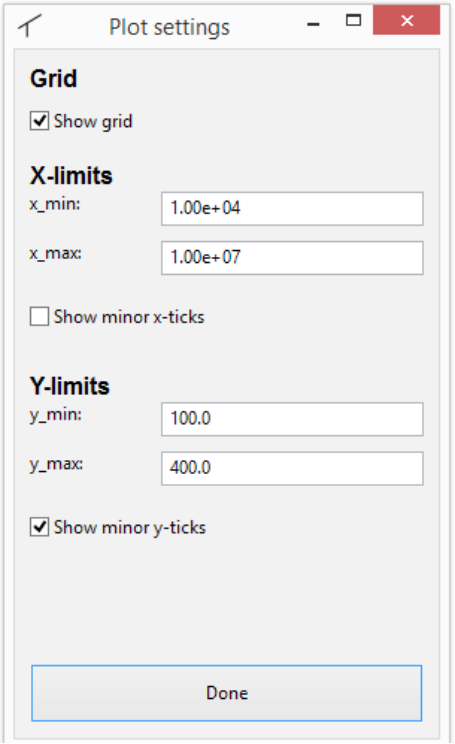
Add/edit manual lines:



Trykker man på knapp "Add/ edit manual lines" (4.) får man opp en meny med oversikt over eventuelle linjer man allerede har lagt inn, med mulighet til å endre eller slette disse. Man kan også legge til nye linjer.

Nye linjer legges til ved å spesifisere navn, stigning og skjæringspunkt (m og c) for en linje som følger formen som står oppgitt under "Add S-N line". Man velger også gyldighetsområde for linjen, dvs. ved hvilke spenning den skal starte og slutte. Når man trykker på "Add" legges linjen til oversikten.

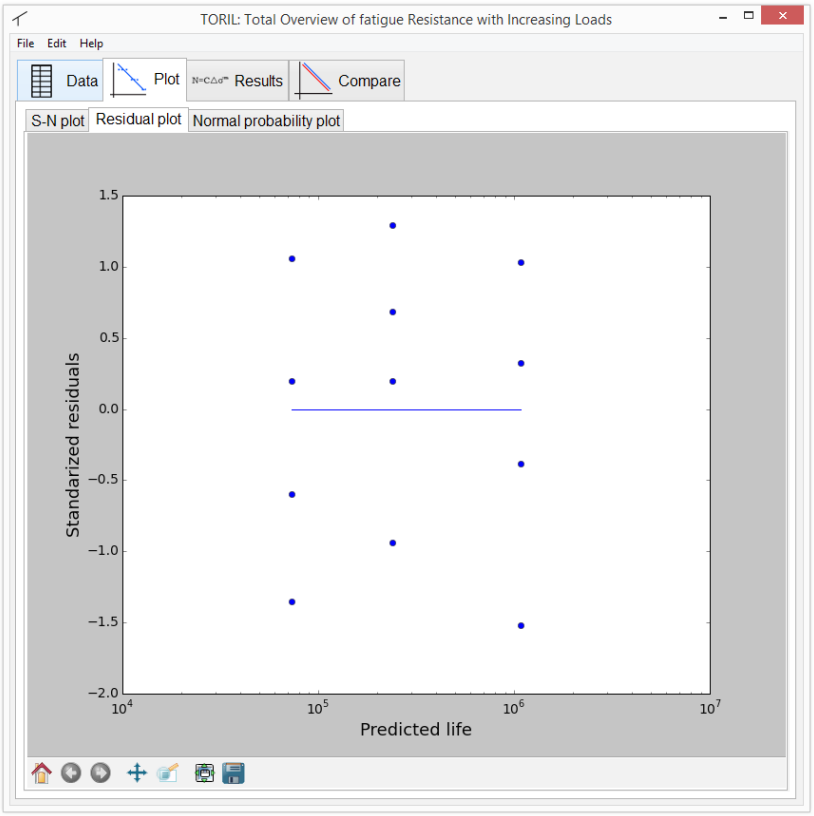
Ønsker man å fjerne eller endre en linje fører man opp nummeret til den aktuelle linjen i feltet nede til høyre og velger "Edit/remove". Linjen forsvinner da fra oversikten, men verdiene blir ført inn i feltene til "Add S-N line". På den måten kan man gjøre ønskede endringer og så legge linjen til på nytt. Endringene registreres først hos resten av programmet når man trykker "Done".



Adjust plot:
Trykker man på "Adjust plot" (5.) kan man gjøre flere endringer på plottet av linjen(e). Man kan velge om det skal vises linjer i bakgrunnen ("Grid") og endre grenseverdiene på X og Y aksen.

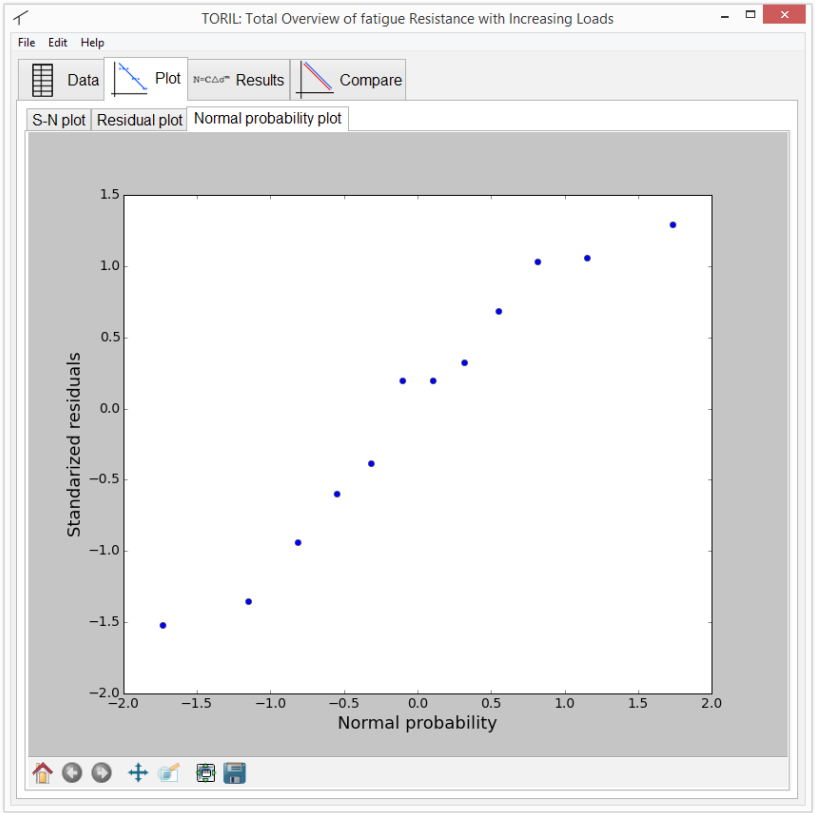
Man kan også velge om man ønsker nummerering mellom de logaritmiske verdiene ("minor ticks"). Disse verdiene vil følge vanlig tallform, mens hovedverdiene fremstår på logaritmisk form.

Endringene registreres når man trykker "Done".



Fanen "Residual plot" vil vise residualplottet for regresjons-linjen, det vil si målepunktenes avvik fra snittet. Dette plottet gir bakgrunn for å vurdere om den rette linjen er en god modell, og hvorvidt målepunktene er uav-hengige av hverandre. Oppfylles dette forventer man at punktene er jevnt fordelt om null-linjen.

Man forventer å få majoriteten av punktene innenfor 1 stan-dardavvik fra 0-linjen, og 95 % innenfor 2 standardavvik.

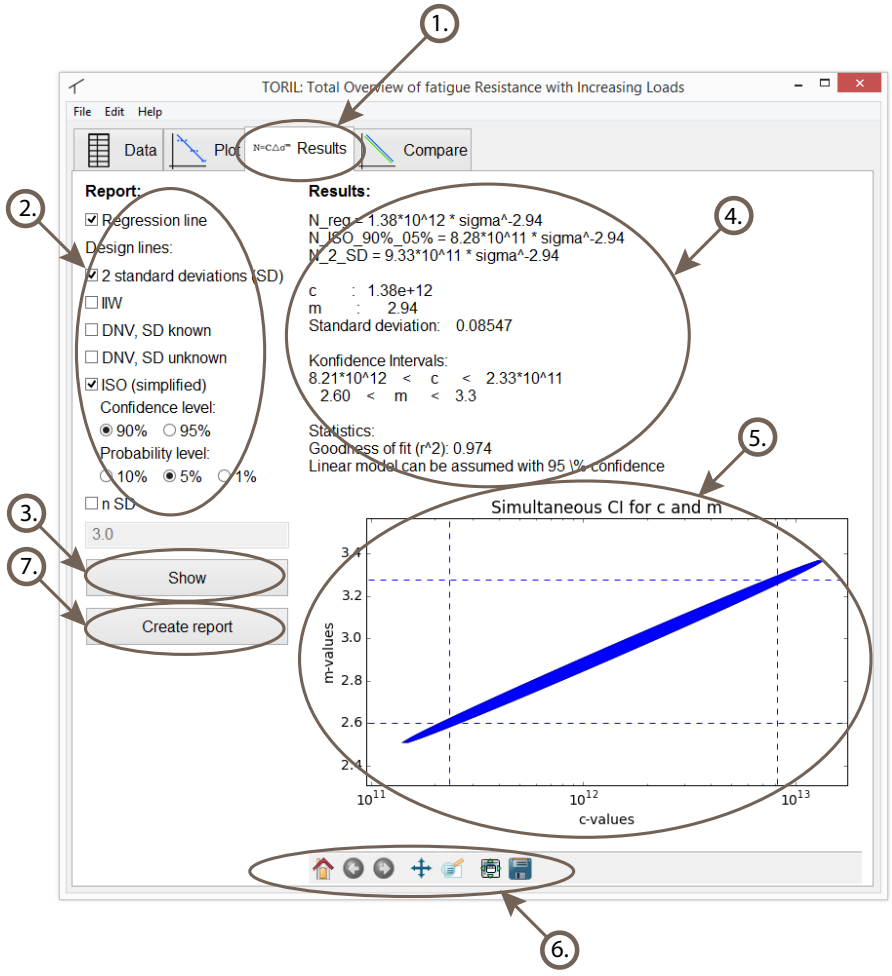


Fanen "Normal probability plot" viser residualene sortert etter størrelse og skal følge en til-nærmet rett linje hvis dataen er normalfordelt. Ved vurdering av plottet legger man størst vekt på sentrale punkter.

4. Evaluere resultater

Under "Results" kan man se linjeverdier og man kan også her velge hvilke designlinjer man ønsker å se ved å huke av de aktuelle linjene og velge "Show". Disse oppdatteres mellom fanene.

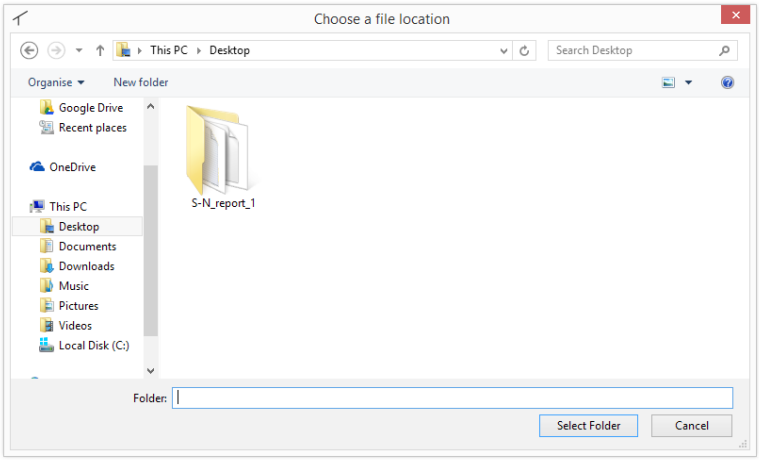
Fanen viser også ulike statistiske verdier av interesse. *c* og *m* svarer til verdiene som angir linjen: $N = c \cdot \Delta\sigma^{-m}$. Konfidensintervallet som oppgis for *c* og *m* svarer til intervallet man forventer at *c* og *m* vil havne innenfor i 95 % av tilfellene hvis testen repeteres. Det er verdt å merke seg at disse verdiene er høyst avhengige av hverandre, så ett simultant konfidensområdet er vist grafisk.



1. Faneplassering
2. Valg for visning
3. Vise resultater
4. Resultater
5. Simultant konfidens-område for c og m
6. Meny for plot
7. Eksportere rapport

"Create report" (7) oppretter en ny mappe med alle figurene, innstillingene fra programmet og en rapportfil i enten PDF eller tekstformat (se kapittel "5. Eksportere rapport"). Rapporten vil inneholde alle resultatene som er huket av når rapporten genereres.

5. Eksportere rapport



Siden eksporteringen lagrer både bilder og filer vil programmet opprette en egen mappe til resultatene. Man velger derfor plasseringen til mappen, som så opprettes og navngis ut i fra prosjektnavnet (se kapittel ”6. Generelle innstillinger”) og vil nummereres fortløpende hvis det eksisterer mapper med det samme navnet.

Man navigerer til ønsket plassering på PCen, f.eks. skrivebordet i bildet over, og velger ”Select Folder”. Står feltet ”Folder” tomt vil mappen plasseres på den plasseringen man er (f.eks. skrivebordet). Trykker man på en av mappene så navnet kommer frem i ”Folder” feltet vil den mappen velges istedenfor.

Alle plottene vil lagres i denne mappen, sammen med en fil med prosjektinnstillingene. Det vil så genereres enten en PDF-fil eller en tekstfil med alle relevante resultater. Velger man PDF rapport vil man også finne tilhørende genereringsfiler (.tex, .aux, .log) i mappen.

Under kapittel ”6. Generelle innstillinger” ser man at de generelle innstillingene gir mulig å bestemme rapportformat, tittel og forfatter, samt spesifisere en introduksjontekst og et sammendrag som inkluderes i rapporten.

Fordelen med en PDF rapport er at den inkluderer alle figurene og gir mer leselige resultater med greske bokstaver og formler. Utformingen av rapporten er fastsatt, men er brukeren kjent med Tex kan .tex filen brukes for å generere egne rapportvarianter.

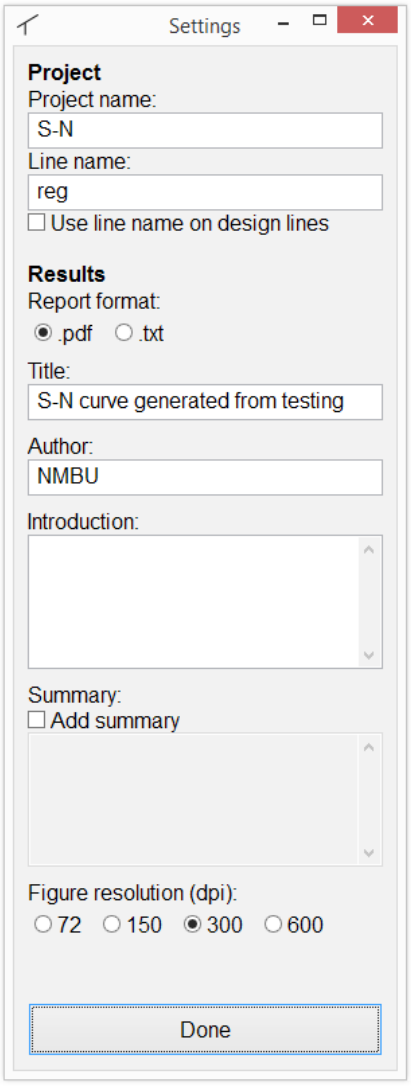
Tekstfilen kan redigeres direkte i etterkant.

Plottene lagres med den oppløsningen man har bestemt i innstillingene, og man kan dermed få en høyere oppløsning enn hva man får ved å lagre bildene direkte (direkte lagring gir 72 dpi).

Programmet gir beskjed når rapporteringen er ferdig.

6. Generelle innstillinger

Under “Settings” i hovedmenyen (se kapittel ”9. Hovedmenyen”) er det mulig å endre generelle innstillinger for prosjektet. Disse innstillingene vil ha betydning for plottingen av resultatene og innholdet og format på rapporten.



Project name: vil brukes ved navngivning av mappe og filer ved eksport av prosjektet.

Line name: vil brukes for å gi navn til regresjonslinen og brukes bl.a. etter N i formelen for linjen.

Use line name on design lines: legger linjenavnet også til designlinjene. Linjenavnet vil da havne før navnet til designlinjen.

Report format: her velger man om man ønsker å få resultatene eksportert som PDF eller som ren tekstfil.

Title: Her kan man velge overskriften på rapporten.

Author: Her settes forfatternavnet som skal brukes i rapporten.

Introduction: Ønskes en introduksjonstekst til rapporten kan dette skrives her.

Add summary: Hvis man ønsker ett sammendrag i rapporten huker man av denne boksen. Det vil da opprettes ett eget avsnitt for dette.

Summary: Her spesifiseres teksten til sammendraget.

Figure resolution: Dette spesifiserer oppløsningen på de eksporterte figurene. 72 dpi er vanlig oppløsning for skjerm, 150 dpi er minimum oppløsning for print, mens 300 dpi er vanlig oppløsning for print. 600 dpi er ikke så vanlig å bruke, men kan brukes om figuren skal forstørres mye.

For å godta innstillingene, velges ”Done”. Krysser man ut vinduet vil ikke endringene registreres.

7. Lagre og importere prosjekter

Man kan lagre og importere prosjekter ved hjelp av ”Open project” og ”Save project” i filmenyen (se kapittel ”9. Hovedmenyen”) .

Ved å lagre prosjektet lagres dataen sammen med innstillinger, både generelle innstillinger og akse-innstillinger, valg av linjer og eventuelle ekstra linjer lagt til plottet. Disse kan senere lastes inn igjen og man vil kunne fortsette fra der man avsluttet.

Det er mulig å lagre prosjektet uten dataverdier, men med innstillinger som navn med mer hvis man ønsker en personlig mal for senere bruk.

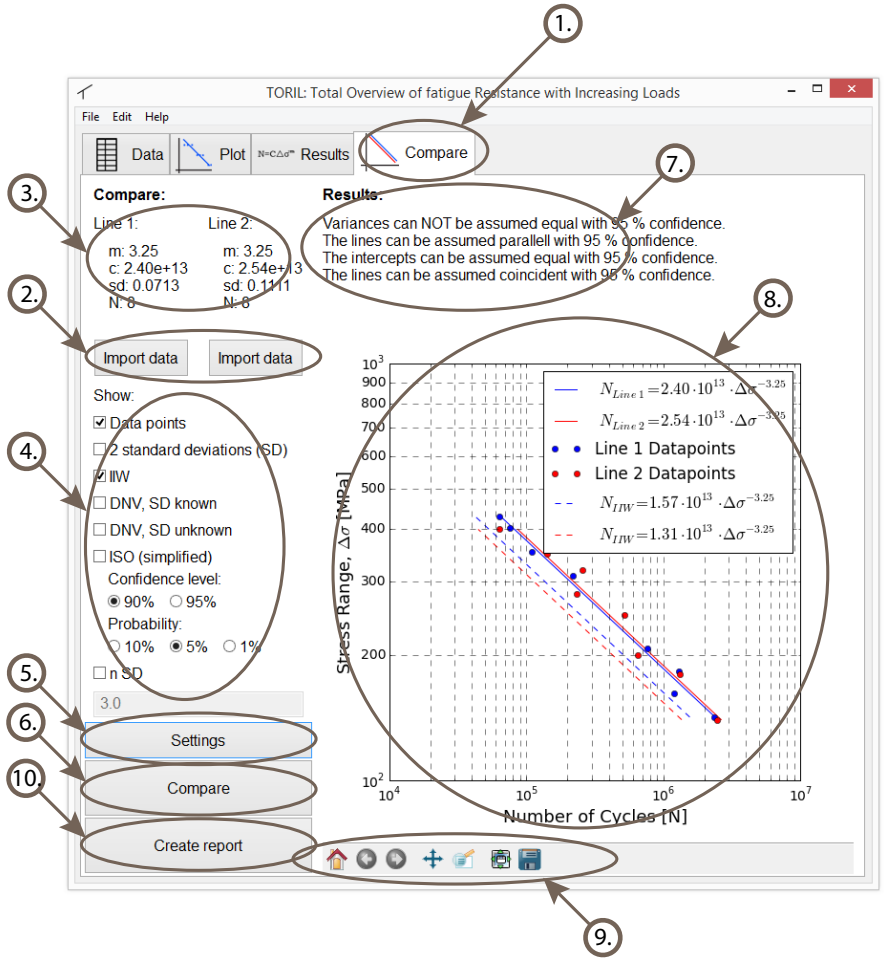
Prosjektinnstillingene lagres også ved eksport av rapport for enkelt å kunne gjøre endringer senere.

Det er mulig å kun hente ut dataverdiene fra filen ved å velge prosjektfilen når man importerer data i datafanen eller i sammenlikningsfanen.

Sammenlikningsfanen er ikke inkludert i lagring av innstillingene.

8. Sammenlikne linjer

Fanen for sammenlikning av linjer er fristilt fra resten av programmet og bruker ikke de generelle innstillingene. Resultatene herfra skrives også til en egen rapport.



1. Faneplassering.
2. Import av linjer.
3. Linjeinfo.
4. Valg for visning.
5. Innstillinger for plot og rapportering.
6. Vise resultater.
7. Resultater.
8. Plot av linjene.
9. Meny for plot.
10. Eksportere rapport.

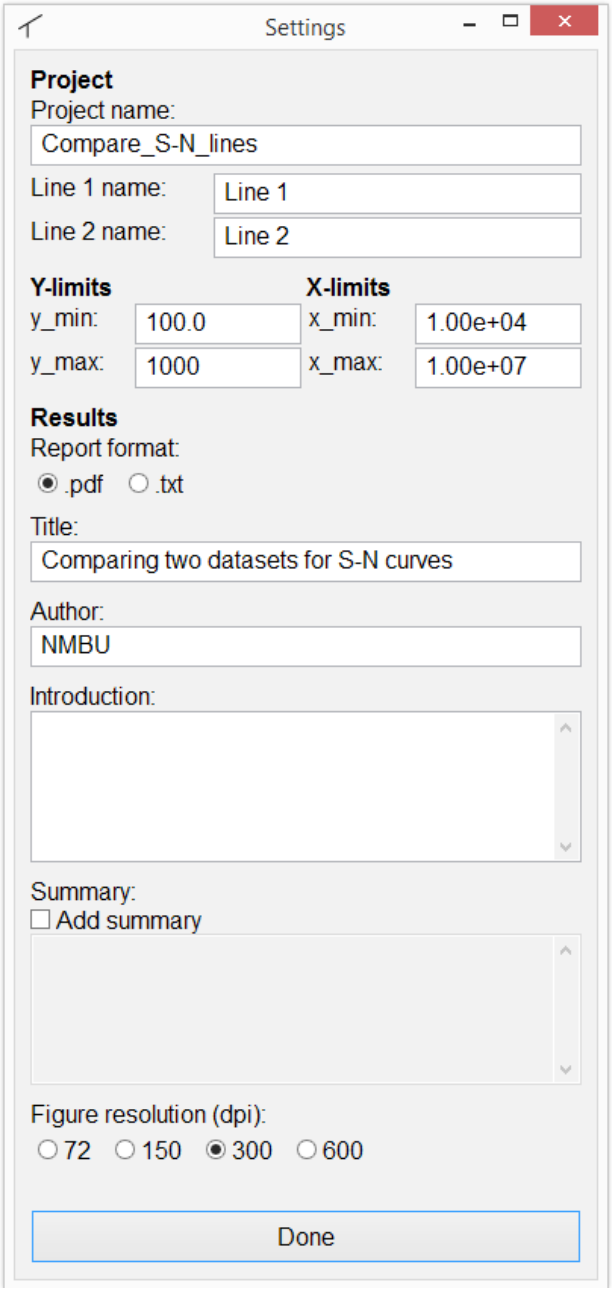
Importen av data følger samme krav som datafanen til format.

Siden indre variasjon i datasettet vil ha stor innvirkning på linjenes designlinjer er det mulig å velge visning av både designlinjer og datapunkter. Disse vises ved å huke av de aktuelle boksene og så trykke på (6): ”Compare”.

Fanen har egne innstillinger som man vil kjenne igjen fra de generelle innstillingene, men har i tillegg verdier for aksene på plottet i tillegg til individuelle navn på linjene. For en oversikt over innstillingsmuligheter, se neste side.

Rapportfunksjonen har også store likhetstrekk med rapporteringen av selve S-N linjen. Omfanget på rapporten er derimot litt mindre og det lagres ikke en prosjektfil, men det lagres en datafil fra hver linje med datapunktene.

Innstillinger



Innstillingene til sammenlikningsfanen har store likhetstrekk med de generelle innstillingene, men disse gjelder kun for sammenlikningsfanen.

Siden man i denne fanen sammenlikner to ulike datasett kan man velge navn fritt for hvert datasett. ”Line 1 name” henviser da til datasettet importert under overskriften ”Line 1” i fanen.

Fanen tilbyr ikke en egen meny for å justere plottet, men tilbyr justering av X og Y-aksene i innstillingsmenyen.

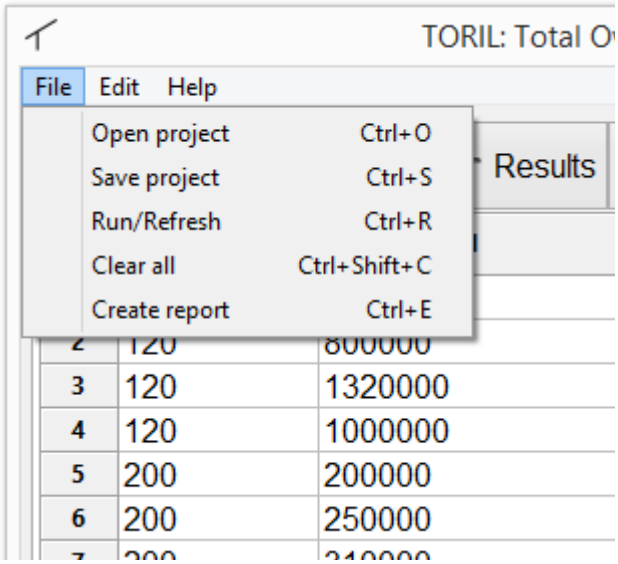
Valgene for resultatene er like som for de generelle innstillingene med unntak av at det her er en annen standardverdi for tittelen.

Endringene registreres når man trykker ”Done”.

9. Hovedmenyen

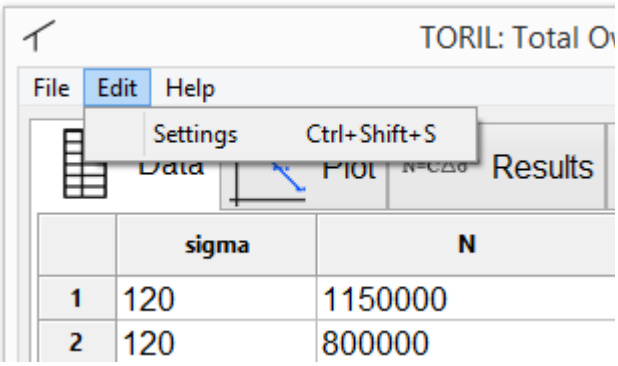
Programmet har en menybar med generelle funksjoner i toppen av vinduet. Her finner man funksjonalitet som er gyldig på tvers av fanene og hjelpefiler. Disse har tilhørende tastatursnarveier for å gjøre bruken av programmet enklere.

Merk at funksjonene i ”File” og ”Edit” gjelder for hoveddelen av programmet, som er å lage S-N kurver fra data, og ikke sammenlikningsfanen.

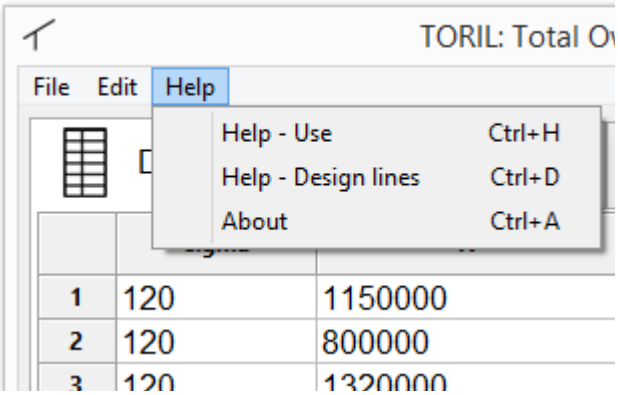


Under ”File” har man mulighet til å åpne og lagre prosjekter (se kapittel ”7. Lagre og importere prosjekter”) og man kan eksportere resultatene til en rapport direkte, uten å gå om resultatfanen.

Det er også mulig å kjøre programmet/oppdatere resultatene. Dette vil oppdatere resultatene med eventuelle nye verdier og/eller valg. Her er tastatursnarveien spesielt nyttig. Her kan man også tilbakestille programmet, dvs. slette data og nullstille innstillingene.



Under ”Edit” finner man de generelle innstillingene (se kapittel ”6. Generelle innstillinger”)



Under ”Help” kan man få en instruksjon på engelsk for bruk av programmet og en oversikt over de ulike designlinjene man kan velge i programmet.

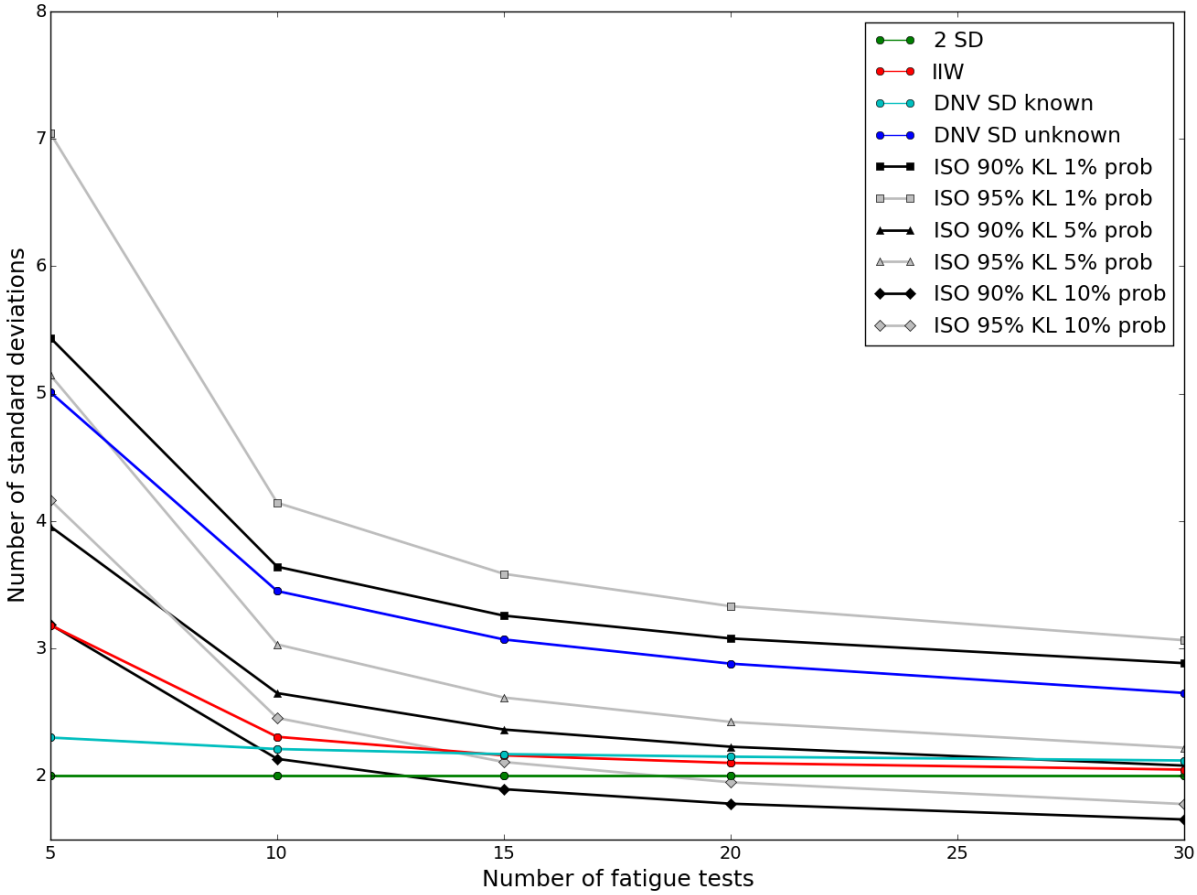
Her finner man også generell informasjon om programmet.

10. Oversikt over designlinjene

Designlinjen baserer seg på at dataen er normalfordelt rundt regresjonslinjen, en linje som tilsvarer 50 % sannsynlighet for brudd, og at man ved å gå langt nok vekk fra linjen kan minske sannsynligheten for brudd betraktelig. Ser man på de rene egenskapene til normalfordelingen så vil en linje 2 standardavvik fra regresjonslinjen gi en 97,7 % sikkerhet mot brudd. Britisk Standard bruker derfor 2 standardavvik som anbefaling, men oppgir også sikkerheter ved bruk av andre antall:

Nominell sannsynlighet for brudd i %	Standardavvik
50	0
31	0,5
16	1,0
2,3	2,0
0,14	3,0

En svakhet med denne metoden er derimot at den ikke tar hensyn til hvor sikkert estimatet av regresjonslinjen er. De andre standardene bruker derfor sikkerheter som er avhengig av antall prøver, noe man kan se illustrert i figuren under.



International Institute of Welding (IIW) bruker en modell som korrigerer både for antall prøvestaver og bruker en Student t-fordeling for å beregne designlinjen. Den samme formelen finner man i en rapport fra Sintef.

Den Norske Veritas (DNV) oppgir en tabell for antall standardavvik man bør bruke og de skiller mellom tester hvor man kan anta kjent standardavvik og ikke. Siden man beregner standardavvik fra testdataen er den statistisk sett ikke kjent, noe som gir behov for store sikkerhetsmarginer. Kan man med ingeniørvurderinger kan si at den stemmer med tidligere resultater og anta at den er kjent blir sikkerhetsmarginene mye mindre. Begge linjene vil nærme seg 2 når antall tester går mot uendelig.

International Standard (ISO) bruker en modell tilsvarende IIW, men da denne bruker en usentral Student t-fordeling stiller de med en egen tabell for disse verdiene. Deres modell baserer seg på at man velger både en pålitelighet til estimeringen (1-probability) og konfidensnivå (Confidence Level) til estimatet. De kommer ikke med noen anbefalinger til sikkerhet, men man kan få ett inntrykk av sikkerheten til de ulike linjene i forhold til de andre standardene i figuren på forrige side. Merk at linjene med 10 % sannsynlighet havner langt under en sikkerhet ekvivalent til 97,7 %.

S-N curve generated from testing

NMBU

May 4, 2015

This is an example of a report generated with the program Toril.

The sample data consists of 12 results from tests taken on three levels 120, 200 and 300 MPa.

1 Input

$\Delta\sigma$	Number
120	1150000
120	800000
120	1320000
120	1000000
200	200000
200	250000
200	310000
200	275000
300	90000
300	76000
300	65000
300	56000

2 Output

S-N line: $N_{reg} = 1.38 \cdot 10^{12} \cdot \Delta\sigma^{-2.94}$

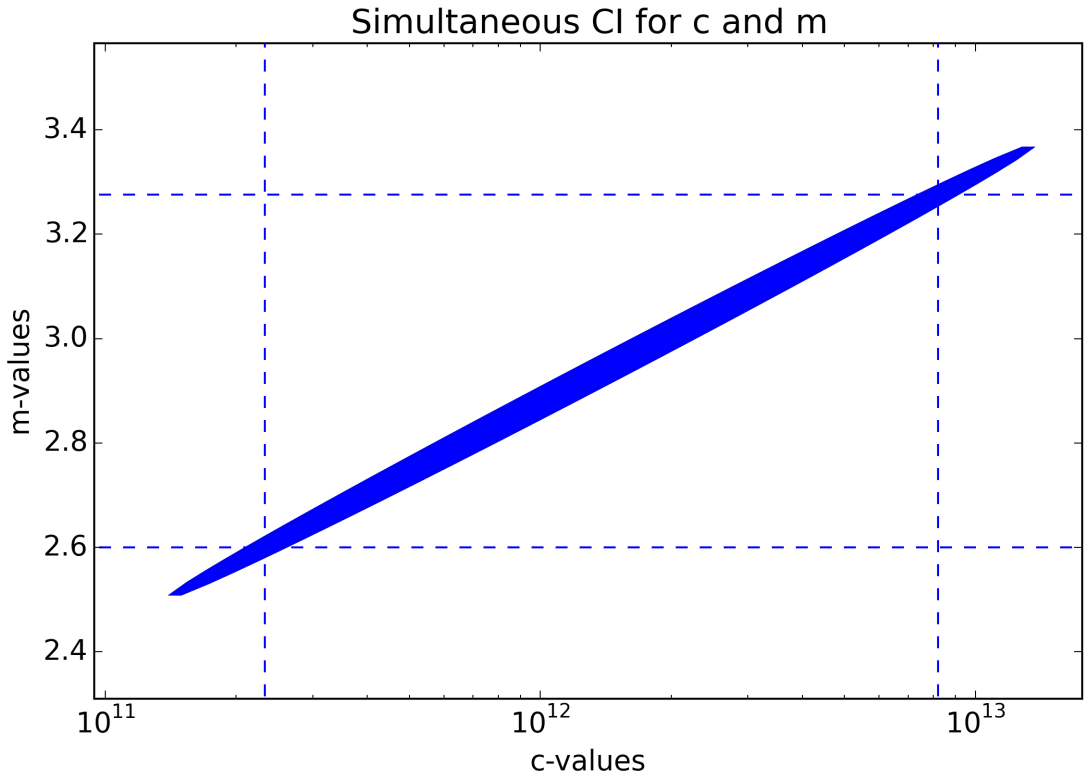
$N_{IIW} = 8.76 \cdot 10^{11} \cdot \Delta\sigma^{-2.94}$

Standard deviation: 0.085

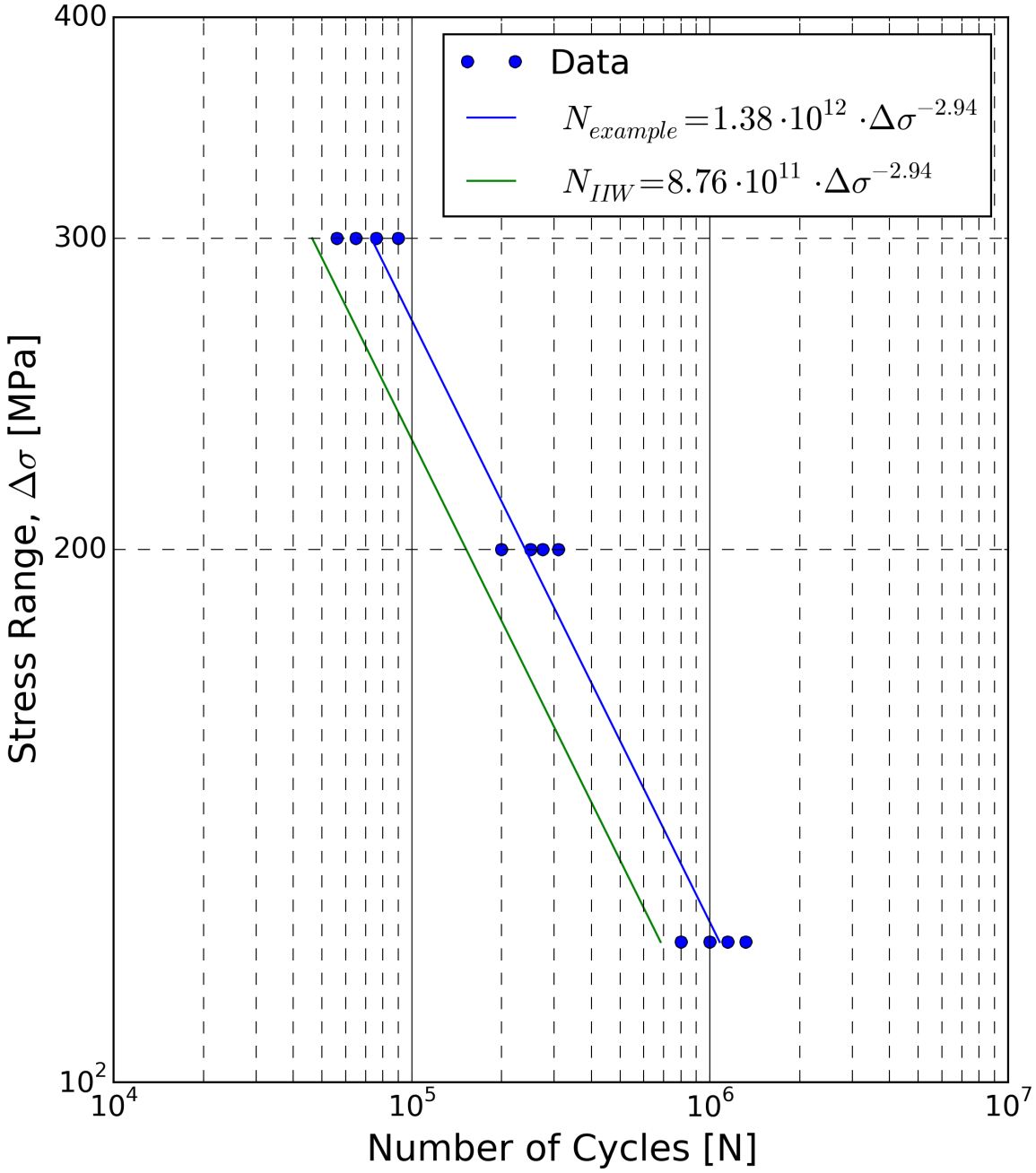
95 % confidence limits for m and c:

$$\begin{aligned} 2.33 \cdot 10^{11} &< c < 8.21 \cdot 10^{12} \\ 2.60 &< m < 3.28 \end{aligned}$$

Simultaneous confidence interval:



2.1 S-N plot

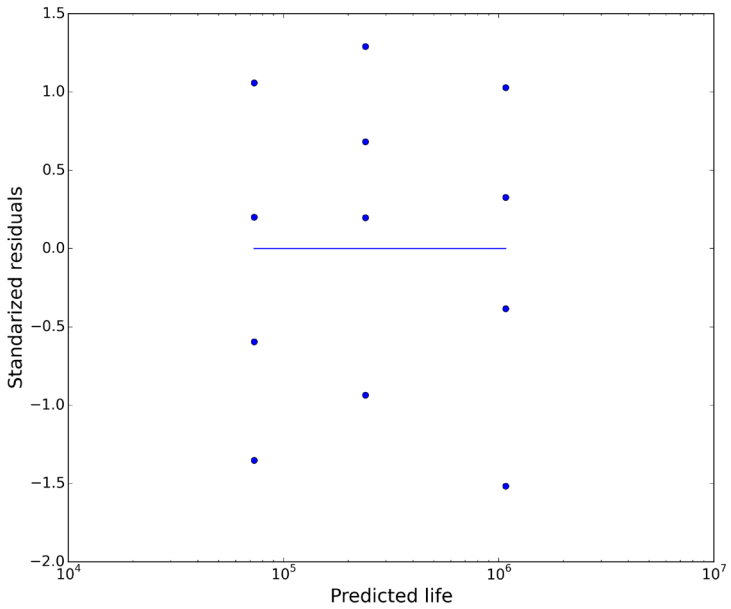

Data
$N_{example} = 1.38 \cdot 10^{12} \cdot \Delta\sigma^{-2.94}$
$N_{IIW} = 8.76 \cdot 10^{11} \cdot \Delta\sigma^{-2.94}$
Stress Range, $\Delta\sigma$ [MPa]
Number of Cycles [N]
400
300
200
10^2
10^4
10^5
10^6
10^7

2.2 Statistical analysis

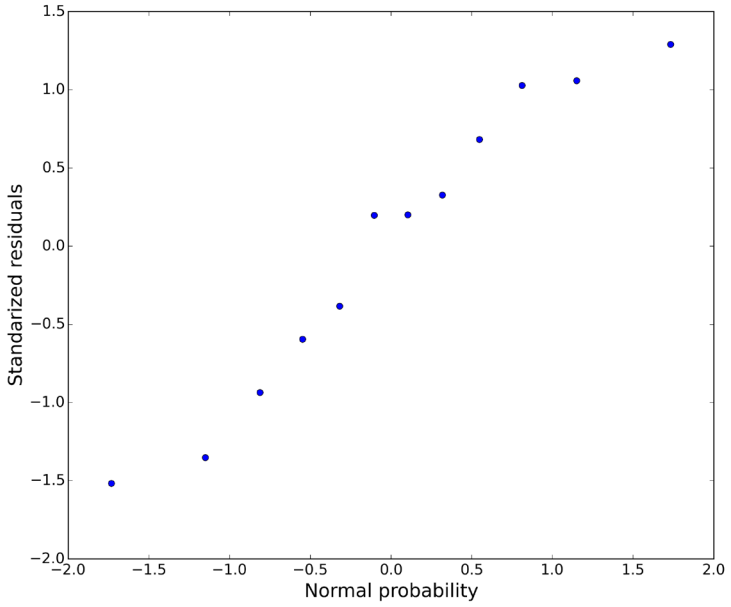
Goodness of fit (r^2): 0.974

Adequacy of a linear model: Linear model can be assumed with 95 % confidence

Residual plot:



Normal probability plot:



2.3 Summary

The results from this example shows a model with a good distribution of residuals and the line scores well on linearity, both from testing and from visual considerations.

IIW is chosen for the design line as it is seen as a god model that takes both the number of tests and the standard deviation into account.

Comparing two datasets for S-N curves

NMBU

May 4, 2015

This example compares two datasets, each with 8 values, with a different degree of variance within the set.

1 Input

Example 1		Example 2	
$\Delta\sigma$	N cycles	$\Delta\sigma$	N cycles
430	63880	400	64000
403	76440	350	141000
353	110500	320	255100
309	219000	280	234600
207	764000	250	520000
183	1307000	200	651000
162	1194000	180	1315000
142	2342000	140	2450000

2 Output

Type	Example 1	Example 2
S-N line	$N_{reg} = 2.40 \cdot 10^{13} \cdot \Delta\sigma^{-3.25}$	$N_{reg} = 2.54 \cdot 10^{13} \cdot \Delta\sigma^{-3.25}$
IIW	$N_{IIW} = 1.57 \cdot 10^{13} \cdot \Delta\sigma^{-3.25}$	$N_{IIW} = 1.31 \cdot 10^{13} \cdot \Delta\sigma^{-3.25}$

2.1 Statistical analysis

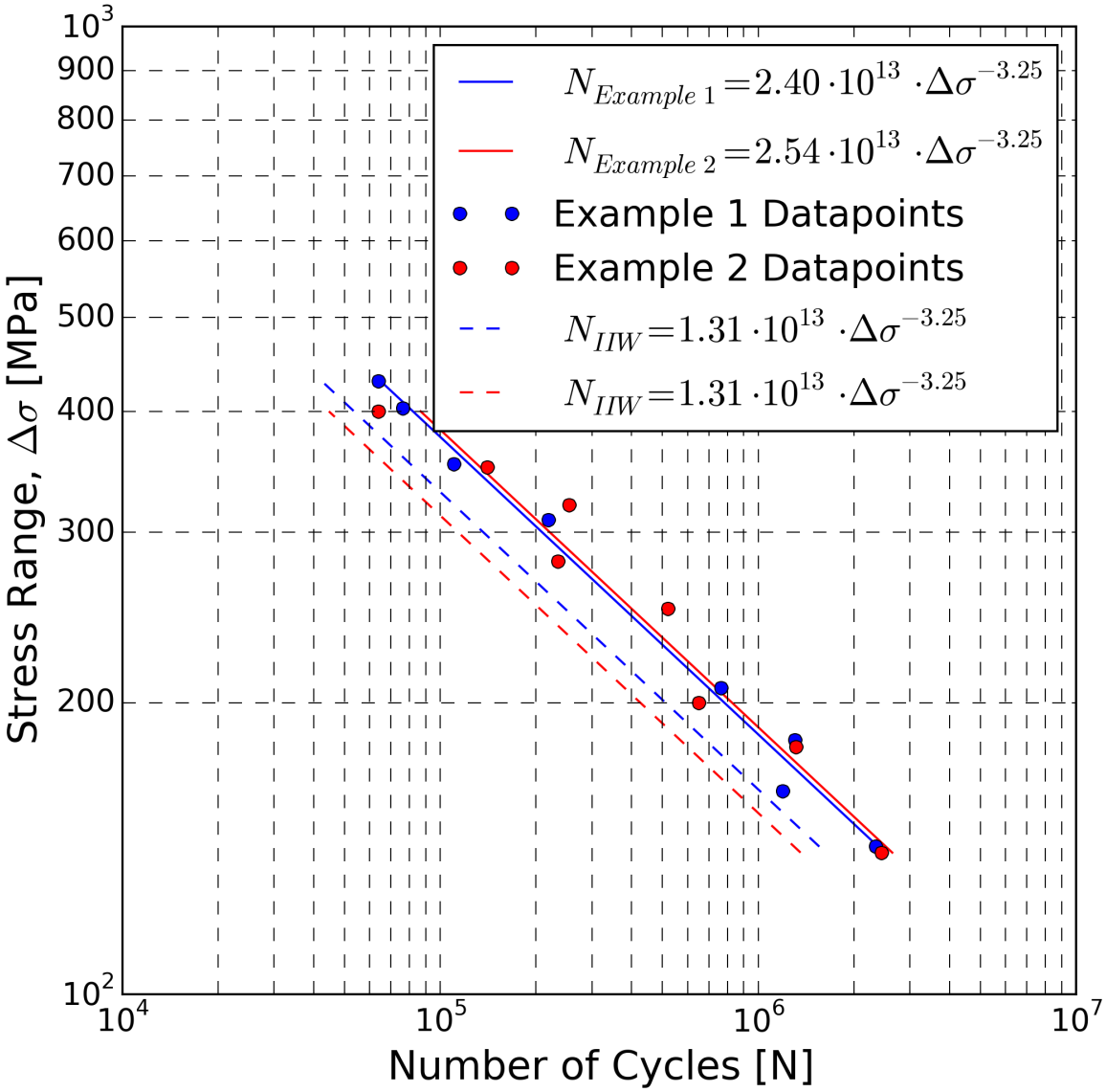
Variances can NOT be assumed equal with 95 % confidence.

The lines can be assumed parallell with 95 % confidence.

The intercepts can be assumed equal with 95 % confidence.

The lines can be assumed coincident with 95 % confidence.

2.2 Plot


$N_{Example\ 1} = 2.40 \cdot 10^{13} \cdot \Delta\sigma^{-3.25}$
$N_{Example\ 2} = 2.54 \cdot 10^{13} \cdot \Delta\sigma^{-3.25}$
Example 1 Datapoints
Example 2 Datapoints
$N_{IIW} = 1.31 \cdot 10^{13} \cdot \Delta\sigma^{-3.25}$
$N_{IIW} = 1.31 \cdot 10^{13} \cdot \Delta\sigma^{-3.25}$
Stress Range, $\Delta\sigma$ [MPa]
Number of Cycles [N]
10^3
900
800
700
600
500
400
300
200
10^2
10^4
10^5
10^6
10^7

2.3 Summary

This is a good example on how important the variance in the dataset is. The highest S-N line, with a greater number of cycles for the same stress, ends up with a design line that is far lower than the line it is compared with.

It is important to note that even if the lines can be assumed coinsident, they may not give the same design line.

Kildekode

- Toril.py
- models.py
- report_generator.py
- setup.py

```python
# -*- coding: utf-8 -*-

__author__ = 'Toril Rygg'
__email__ = 'torilrygg@gmail.com'

import wx
import wx.grid
import os
import time
import math as m

import matplotlib
from matplotlib.figure import Figure
from matplotlib.ticker import FormatStrFormatter
from matplotlib.backends.backend_wxagg import FigureCanvasWxAgg as FigureCanvas
from matplotlib.backends.backend_wxagg import NavigationToolbar2WxAgg as NavigationToolbar

from models import *
from report_generator import *

import sys
reload(sys)
sys.setdefaultencoding('utf-8')
sys.stdout = open('my_stdout.log', 'w')
sys.stderr = open('my_stderr.log', 'w')

font = {'weight': 'normal',
        'size': 14}

matplotlib.rc('font', **font)

SETTINGS_TYPE = {'project_name': 'str',
                 'line_name': 'str',
                 'use_line_name': 'bool',
                 'report_format': 'str',
                 'title': 'str',
                 'author': 'str',
                 'intro': 'str',
                 'use_summary': 'bool',
                 'summary': 'str',
                 'dpi': 'str'}

PLOT_SETTINGS_TYPE = {'show_grid': 'bool',
                      'show_minor_x': 'bool',
                      'show_minor_y': 'bool',
                      'x_lim': 'tuple',
                      'y_lim': 'tuple'}


# noinspection PyUnusedLocal
class MainFrame(wx.Frame):
    def __init__(self, parent=None):
        super(MainFrame, self).__init__(parent,
                                        title='TORIL: Total Overview of fatigue Resistance with Increasing Loads',
                                        size=(850, 850))
        self.default_font = wx.Font(12, wx.DEFAULT, wx.NORMAL, wx.NORMAL)
        self.make_menu()
        self.Bind(wx.EVT_CLOSE, self.on_close)

        self.SetFont(self.default_font)

        self.settings = {'project_name': 'S-N', 'line_name': 'reg', 'use_line_name': False,
                         'report_format': '.pdf', 'title': 'S-N curve generated from testing',
                         'author': 'NMBU', 'intro': '', 'use_summary': False, 'summary': '', 'dpi': '300'}

        self.x_lim = (1e4, 1e7)
        self.y_lim = (1e2, 1e3)

        self.name = None    # manual lines data
        self.c = None
        self.m = None
        self.sigma_min = None
        self.sigma_max = None

        icon = wx.Icon('graphics/T_logo.ico', wx.BITMAP_TYPE_ICO)
        self.SetIcon(icon)

        self.panel = wx.Panel(self)
        self.notebook = NotebookStructure(self, self.panel)
        self.data_tab = self.notebook.get_data_tab()
        self.plot_tab = self.notebook.get_plot_tab()
        self.results_tab = self.notebook.get_results_tab()

        vsizer = wx.BoxSizer(wx.VERTICAL)
        vsizer.Add(self.notebook, 1, wx.ALL | wx.EXPAND, 5)

        self.panel.SetSizer(vsizer)
        self.Centre()
```

```
        self.Show()

    def make_menu(self):
        menubar = wx.MenuBar()
        file_menu = wx.Menu()
        edit_menu = wx.Menu()
        help_menu = wx.Menu()

        menubar.Append(file_menu, '&File')
        menubar.Append(edit_menu, '&Edit')
        menubar.Append(help_menu, '&Help')

        file_menu.Append(wx.ID_OPEN, '&Open project\tCtrl+O')
        file_menu.Append(wx.ID_SAVE, '&Save project\tCtrl+S')
        file_menu.Append(wx.ID_EXECUTE, '&Run/Refresh\tCtrl+R')
        file_menu.Append(wx.ID_CLEAR, '&Clear all\tCtrl+Shift+C')
        file_menu.Append(wx.ID_FILE, '&Create report\tCtrl+E')
        edit_menu.Append(wx.ID_SETUP, '&Settings\tCtrl+Shift+S')
        help_menu.Append(wx.ID_HELP, '&Help - Use\tCtrl+H')
        help_menu.Append(wx.ID_HELP_CONTENTS, '&Help - Design lines\tCtrl+D')
        help_menu.Append(wx.ID_ABOUT, '&About\tCtrl+A')

        menubar.SetFont(self.default_font)

        self.Bind(wx.EVT_MENU, self.on_load_project, id=wx.ID_OPEN)
        self.Bind(wx.EVT_MENU, self.on_save_project, id=wx.ID_SAVE)
        self.Bind(wx.EVT_MENU, self.on_run, id=wx.ID_EXECUTE)
        self.Bind(wx.EVT_MENU, self.on_clear, id=wx.ID_CLEAR)
        self.Bind(wx.EVT_MENU, self.on_report, id=wx.ID_FILE)
        self.Bind(wx.EVT_MENU, self.on_settings, id=wx.ID_SETUP)
        self.Bind(wx.EVT_MENU, self.on_design_help, id=wx.ID_HELP_CONTENTS)
        self.Bind(wx.EVT_MENU, self.on_help, id=wx.ID_HELP)
        self.Bind(wx.EVT_MENU, self.on_about, id=wx.ID_ABOUT)
        self.SetMenuBar(menubar)

    def on_load_project(self, event=None):
        wildcard = "Accepted file formats (*.txt)|*.txt"
        dialog = wx.FileDialog(self, 'Import project file', wildcard=wildcard, style=wx.OPEN)
        if dialog.ShowModal() == wx.ID_OK:
            path = dialog.GetPath()
            self.notebook.set_update(True)
            self.load_project(path)
            dialog.Destroy()

    def load_project(self, path):
        file_name, file_format = os.path.splitext(path)
        new_data = True
        sigma_values = []
        n_values = []
        settings = {}
        actives = {}
        plot_settings = {}
        manual_lines = []
        self.name = None    # manual lines data
        self.c = None
        self.m = None
        self.sigma_min = None
        self.sigma_max = None
        n_sd = None

        is_settings_file = False
        is_data = False
        is_settings = False
        is_actives = False
        is_plot = False
        is_lines = False
        with open(path, 'r') as infile:
            for data in infile.readlines():
                if new_data:
                    if 'Project settings:' in data:
                        is_settings_file = True
                    elif 'sigma' in data.lower():
                        is_data = True
                    elif 'plot settings' in data.lower():
                        is_plot = True
                    elif 'settings' in data.lower() and 'project' not in data.lower():
                        is_settings = True
                    elif 'actives' in data.lower():
                        is_actives = True
                    elif 'manual line' in data.lower():
                        is_lines = True
                    new_data = False
                else:
                    if len(data) <= 1 or data == '\n':    # New block of data
                        new_data = True
                        is_data = False
                        is_settings = False
                        is_actives = False
```

```
                    is_plot = False
                    is_lines = False
                    if self.name:
                        manual_lines.append(ManualLine(self.name, self.c, self.m, self.sigma_min, self.sigma_max))
                        self.name = None
                else:
                    if is_data:
                        sigma, n = data.split(self.data_tab.delimiter[file_format])
                        sigma_values.append(eval(sigma))
                        n_values.append(eval(n))
                    elif is_settings:
                        split = data.split(':')
                        if len(split) > 2:
                            key = split[0]
                            value = ''
                            for element in split[1:]:
                                value += element
                        else:
                            key, value = data.split(':')
                        if SETTINGS_TYPE[key] == 'str':
                            value = value.strip()
                            if key == 'summary' or key == 'intro':
                                settings[key] = eval(value)
                            else:
                                settings[key] = value
                        else:
                            settings[key] = eval(value)
                    elif is_actives:
                        key, value = data.split(':')
                        if key == 'n_sd':
                            n_sd = value
                        else:
                            actives[key] = (value.strip() == 'True')
                    elif is_plot:
                        key, value = data.split(':')
                        if PLOT_SETTINGS_TYPE[key] == 'str':
                            plot_settings[key] = value.strip()
                        else:
                            plot_settings[key] = eval(value)
                    elif is_lines:
                        key, value = data.split(':')
                        if key == 'name':
                            self.name = value.strip()
                        elif key == 'c':
                            self.c = eval(value)
                        elif key == 'm':
                            self.m = eval(value)
                        elif key == 'sigma':
                            sigma = eval(value)
                            self.sigma_min = sigma[0]
                            self.sigma_max = sigma[1]
                    else:
                        new_data = True
        if is_settings_file:
            self.data_tab.set_data(sigma_values, n_values)
            self.set_settings(settings)
            self.set_line_label(settings['line_name'])
            self.set_line_use_label(settings['use_line_name'])
            sn_tab = self.notebook.get_sn_tab()
            sn_tab.set_axes_settings(plot_settings)
            sn_tab.set_manual_lines(manual_lines)
            self.notebook.set_n_sd(n_sd)
            self.notebook.set_checked(actives)
            self.data_tab.on_calculate()
        else:
            self.on_error('Settings file not recognized')

    def on_save_project(self, event=None):
        wildcard = "Accepted file formats (*.txt)|*.txt"
        dialog = wx.FileDialog(self, 'Save as', wildcard=wildcard, style=wx.FD_SAVE)
        if dialog.ShowModal() == wx.ID_OK:
            path = dialog.GetPath()
            self.save_project(path)
            dialog.Destroy()

    def save_project(self, path):
        file_format = '.txt'
        sigma_values, n_values = self.data_tab.get_data()
        header = 'Sigma' + self.data_tab.delimiter[file_format] + 'N'
        with open(path, 'w') as infile:
            infile.write('Project settings: \n \n')
            infile.write(header)
            for sigma, N in zip(sigma_values, n_values):
                infile.write('\n' + str(sigma) + self.data_tab.delimiter[file_format] + str(N))
            infile.write('\n\n')
            infile.write('Settings')
            for key, value in self.settings.items():
                if key == 'summary' or key == 'intro':
```

```
265                 infile.write('\n' + key + ':' + repr(value))
266             else:
267                 infile.write('\n' + key + ':' + str(value))
268         infile.write('\n\n')
269         infile.write('Actives')
270         for key, value in self.notebook.active_methods.items():
271             infile.write('\n' + key + ':' + str(value))
272         for key, value in self.notebook.iso_options.items():
273             infile.write('\n' + key + ':' + str(value))
274         infile.write('\n')
275         infile.write('n_sd:' + str(self.notebook.get_n_sd()))
276         infile.write('\n\n')
277         sn_tab = self.notebook.get_sn_tab()
278         infile.write('Plot settings')
279         for key, value in sn_tab.get_axes_settings().items():
280             infile.write('\n' + key + ':' + str(value))
281         infile.write('\n\n')
282         for line in sn_tab.get_manual_lines():
283             infile.write('Manual line')
284             sigma = list(line.sigma_values)
285             infile.write('\n' + 'name' + ':' + line.name)
286             infile.write('\n' + 'c' + ':' + str(line.c))
287             infile.write('\n' + 'm' + ':' + str(line.m))
288             infile.write('\n' + 'sigma' + ':' + str(sigma))
289             infile.write('\n\n')
290
291     def on_run(self, event=None):
292         self.data_tab.on_calculate(event)
293
294     def on_clear(self, event=None):
295         if wx.MessageBox('This will delete all the data and reset all setting. Are you sure you want to continue?',
296                          'Clear all?',
297                          wx.YES_NO | wx.YES_DEFAULT) == wx.YES:
298             self.notebook.reset()
299
300     def on_report(self, event=None):
301         self.results_tab.on_report()
302
303     def on_settings(self, event=None):
304         SettingWindow(self)
305
306     def set_settings(self, dictionary):
307         self.settings = dictionary
308
309     def set_line_label(self, label):
310         self.notebook.get_line().set_label(label)
311
312     def set_line_use_label(self, boolean):
313         self.notebook.get_line().set_use_label(boolean)
314
315     def get_x_lim(self):
316         return self.x_lim
317
318     def set_x_lim(self, x_lim):
319         self.x_lim = x_lim
320
321     def get_y_lim(self):
322         return self.y_lim
323
324     def set_y_lim(self, y_lim):
325         self.y_lim = y_lim
326
327     def get_settings(self):
328         return self.settings
329
330     def on_help(self, event=None):
331         message = '''The program consists of two parts:
332 - Creating S-N curves from test data end evaluate the results (Tab 1-3).
333 - Comparing two sets of test data (Tab 4).
334 Below follows an overview of the functionality and use of the different tabs, and the program in general.
335
336 CREATING AN S-N CURVE
337 ***** DATA *****
338 Start with adding data in the first tab: Data. This can be done manually, or by importing\
339 data from a .txt og .csv file with the "Import data" button. In the .txt file, the data must be separated by tabs, \
340 and in the .csv file with ";". The program can also retrieve the nececary data from an earlier generated \
341 settings file. The data can be saved with the "Save data" button for later use.
342
343 After the data is added, pressing "Calculate" will make all the nececary calculation and prepare the plots.
344
345 It is also possible to export as a report immediately through the "File" menu.
346
347 ***** PLOT *****
348 The plot tab consists of three sub tabs: S-N plot, Residual plot and Normal probability plot:
349
350 ----- S-N Plot -----
351 This is the main plot, where the results are directly visualized. The default option is to show the data points and \
352 the regression line, but this can be changed by ticking and unticking the boxes under "Show:".
```

The design lines that are chosen when the report is created, will be included in the report. For more information \
on the differences between the design lines, see "Help -> Help - Design lines".

Press the "Show" button after changes in selection to update the plot.

It is also possible to add extra lines to the plot with the "Add/edit manual lines" button. \
These lines will be shown with the rest of the lines and can be included in the report. \
Pressing "Add/edit manual lines" will open a new window where it is possible to add manual lines by giving \
c and m values for it, as well as defining a name and a stress range. This window will also give an overview of all \
the manual lines and it is possible to remove or change the lines with the "Edit/remove" button. \
The program will then remove the line (with the number specified in the bottom of the window) from the overview \
and load the details in the "Add S-N line" area for possible editing.

To change the plot itself (limits, grid etc.), press the "Adjust plot" button and change the values in the popup menu.

----- Residual plot -----
The residual plot shows how much each data point differ from the regression line. The values are corrected for the \
standard deviation, so the majority of values should be within one standard deviation, and 95 % within 2 \
standard deviatons from the zero-line.

The points should be equally distributed around the horizontal zero-line. If this is not the case, this could indicate \
a non linear model or that the values are not statistically independent.

----- Normal probability plot -----
Here, the residuals are sorted by value and plotted against equally distributed points. This plot should form \
an approximately straight line if the data is normally distributed. The central values are of greatest importance \
when considering the results.

***** RESULTS *****
This tab shows the results as text and includes statistical values for the line. It will also show the design line(s) \
chosen under "Report" to the left, as well as the manual lines if checked. \
These boxes are linked with the "Plot" values and will update between tabs.

The "Show" button will update the results with the chosen lines, and the "Create report" button will generate a report \
folder with the plots, program settings and a report file. By default, the report file is a PDF with the plots \
included, but it is also possible to choose a .txt file in the "File -> Settings" menu. The report will include \
everything that is checked at that moment.

c and m corresponds to the values in the formula N = c*sigma^-m, and the standard deviation is a measure of the \
amount of variation in the data set.

c and m has corresponding 95 % confidence intervals, meaning the interval where the values of c and m are expected to \
be included 95 % of the time if the experiment is repeated. It is worth noting that the values of c and m are highly \
dependent, and a simultaneous CI for c and m are shown with a figure.

COMPARING DATASETS
***** COMPARE *****
Here it is possible to compare two sets of test data. If a line has already been added in the "Data" tab, this will \
be the default line for Line 1. Both lines can be added with the corresponding "Import data" button, and the \
requirements for the file is the same as for the "Data" tab.

By pressing the "Compare" button, the program will check if the variances and intercepts can be assumed equal, and \
if the lines can be assumed parallel. The lines will also be plotted visually against each other. \
It is also possible to show the design lines since they are highly dependent on variance in the dataset.

This tab does not use the same settings as the rest of the program and it is therefore a "Settings" button \
where the name of the lines can be defined, as well as the plot limits and settings for reporting. These settings \
are very similar to the general settings from the menubar.

MENUBAR
There are several menu options in the top menubar:

***** File *****
Here it is possible to open and save project settings. Saving the project will save both the data currently \
added, along with all the setting options in the program, included checked boxes. The Compare tab is not included.

You also have an universal "Run/Refresh" option, that will update all the values. The keyboard shortcut can be \
especially useful.

It is also possible to clean out the data and reset the settings with "Clear all".

You can also create a report directly, using all the settings and checked values that are active at \
that point.

***** Edit *****
Here, you will find the "Settings" menu.

In "Settings", it is possible to choose project name (name base for all saved files and folders) \
and name for the regression line. If "Use line name on design lines" is checked, the name will also \
appear before the name of the design lines in plots and more.

The "Report" part of the settings are especially usefull, as the title and author can be set for the \
report file, and it is possible to add an introduction and a summary to the report.

***** Help *****

```
Here, you will find this help text, information about the differences between the design lines, and \
information about the program.

'''
        PopupWindow(self, message, 'Help:', (800, 600))

    def on_design_help(self, event=None):
        message = ""The design line is based on the property of the normal distribution of the data around \
the regression line. The regression line itself gives a 50 % probability of failure, but the probability \
decreases substantially when you move away from the line and 2 standard deviations gives a 97.7 % survival rate \
for the design line. Below is an overview of probability of failure vs. standard deviations.

Probability of failure     Number of standard deviations
        50                      0.0
        31                      0.5
        16                      1.0
        2.3                     2.0
        0.14                    3.0

This is however only valid when you have a substantial number of tests. Some of the standards have different \
ways of dealing with this:

IIW (IIW-XIII-WG1-114-03) use a simplified statistical method based on a Student t-distribution for calculating the \
design line, one that takes \
the number of tests into account. This is valid for values near the center of the data set, or if you can \
say the spread in the data set is equal the central spread all over.

DNV (DNV-RP-C203) has two sets of correction factors, both decreasing with the number of tests and approaching 2. \
The most conservative is a purely statistical method, where the standard deviation is seen as previously unknown, \
and gives as a factor of well over 3 standard deviations away from the mean line if you have few samples. \
The other option is if the user, by engineering judgement and previous experience, \
can say that the standard deviation is known. This gives a less conservative line, and is closer to 2 standard \
deviations.

ISO (ISO 12107) uses a method similar to IIW, but their method is based on a non-central Student t-distribution. They, \
therefore, give a corresponding statistical table to their formula. \
This table is limited to 25 degrees of freedom. More than 27 samples will therefore not give a smaller correction. \
The table asks for both probability (10, 5 or 1%) and confidence level(90 or 95 %). The probability corresponds \
to the reliability of the prediction (100% - probability) and the confidence level is the confidence of the \
reliability statement.
ISO's formula is not linear, but using the same simplification as IIW, the line follows the normal form for a design \
line.

British Standard (BS 7608:2014) recommends 2 standard deviations, but are open to using a smaller factor if a \
fracture is of low consequence, or a larger factor for a smaller probability of failure. This does not correct for \
the number of tests.
    '''
        PopupWindow(self, message, 'Design lines:', (800, 600))

    def on_about(self, event=None):
        message = ""This program was made by Toril Fjeldaas Rygg as a part of her master thesis at NMBU spring 2015.

This program is free of use for everyone at NBMU, but may not be distributed outside of campus without \
the authors permission.

The user of this program is responsible for using it correctly and that the results are correct.

A more thorough overview of the theoretical background of this program can be found in the master thesis.

This program is written in Python and used wxPython for the graphical user interface.
The PDF report is made with MixTeX and the python script is converted to exe with py2exe.

'''
        PopupWindow(self, message, 'About:', (700, 500))

    def on_error(self, error_message):
        PopupWindow(self, error_message, 'WARNING:', (300, 200))

    def on_close(self, event):
        if wx.MessageBox('Are you sure you want to exit the application?',
                         'Exit application',
                         wx.YES_NO | wx.YES_DEFAULT) != wx.YES:
            return
        self.Destroy()


class NotebookStructure(wx.Notebook):
    METHODS_CHECKED = {'reg_line_cb': True, 'design_2_sd_cb': False, 'design_iiw_cb': False,
                       'design_dnv_known_cb': False, 'design_dnv_unknown_cb': False, 'design_iso_cb': False,
                       'design_n_sd_cb': False}
    ACTIVE_METHODS = {'reg_line': True, 'design_2_sd': False, 'design_iiw': False,
                      'design_dnv_known': False, 'design_dnv_unknown': False, 'design_iso': False,
                      'design_n_sd': False}
    ISO_OPTIONS = {'conf_90': True, 'conf_95': False, 'prob_10': False, 'prob_05': True, 'prob_01': False}

    N_SD = 3.0

    def __init__(self, parent, panel):
```

```
529         wx.Notebook.__init__(self, panel, id=wx.ID_ANY, style=wx.BK_DEFAULT)
530         self.parent = parent
531         self.delimiter = {'.txt': '\t', '.csv': ';'}
532 
533         self.line = Line()
534         self.methods_checked = self.METHODS_CHECKED.copy()
535         self.active_methods = self.ACTIVE_METHODS.copy()
536         self.iso_options = self.ISO_OPTIONS.copy()
537 
538         self.n_sd = self.N_SD
539         self.use_manual_lines = False
540 
541         self.manual_lines = []
542 
543         self.sigma_values = []
544         self.n_values = []
545 
546         self.need_update = False  # If the results needs to be updated
547 
548         self.data_tab = DataTab(self)
549         self.plot_tab = PlotTab(self)
550         self.results_tab = ResultsTab(self)
551         self.compare_tab = CompareTab(self)
552         self.AddPage(self.data_tab, 'Data')
553         self.AddPage(self.plot_tab, 'Plot')
554         self.AddPage(self.results_tab, 'Results')
555         self.AddPage(self.compare_tab, 'Compare')
556 
557         image_list = wx.ImageList(40, 40)  # size of image
558         img_data = image_list.Add(wx.Bitmap('graphics/data.png', wx.BITMAP_TYPE_PNG))
559         img_plot = image_list.Add(wx.Bitmap('graphics/plot.png', wx.BITMAP_TYPE_PNG))
560         img_results = image_list.Add(wx.Bitmap('graphics/results.png', wx.BITMAP_TYPE_PNG))
561         img_compare = image_list.Add(wx.Bitmap('graphics/compare.png', wx.BITMAP_TYPE_PNG))
562 
563         self.AssignImageList(image_list)
564 
565         self.SetPageImage(0, img_data)
566         self.SetPageImage(1, img_plot)
567         self.SetPageImage(2, img_results)
568         self.SetPageImage(3, img_compare)
569 
570     def set_checked(self, active_methods):
571         for key, value in active_methods.items():
572             if 'conf_' in key or 'prob_' in key:
573                 self.iso_options[key] = value
574             else:
575                 self.methods_checked[key + '_cb'] = value
576                 self.active_methods[key] = value
577         self.plot_tab.update_checkbox()
578         self.plot_tab.update_radio_btn()
579         self.results_tab.update_checkbox()
580         self.results_tab.update_radio_btn()
581 
582     def get_iso_conf(self):
583         for key, value in self.iso_options.iteritems():
584             if 'conf' in key and value:
585                 parts = key.split('_')
586                 return parts[1]
587 
588     def get_iso_prob(self):
589         for key, value in self.iso_options.iteritems():
590             if 'prob' in key and value:
591                 parts = key.split('_')
592                 return parts[1]
593 
594     def get_iso_options(self):
595         return self.iso_options
596 
597     def get_manual_lines(self):
598         return self.manual_lines
599 
600     def set_manual_lines(self, lines):
601         self.manual_lines = lines
602 
603     def set_use_manual_lines(self, boolean):
604         self.use_manual_lines = boolean
605 
606     def get_use_manual_lines(self):
607         return self.use_manual_lines
608 
609     def get_data_tab(self):
610         return self.data_tab
611 
612     def save_data(self, path):
613         self.data_tab.save_data(path)
614 
615     def get_plot_tab(self):
616         return self.plot_tab
```

```
617
618     def get_sn_tab(self):
619         return self.plot_tab.get_sn_tab()
620
621     def save_plots(self, path):
622         self.plot_tab.save_plots(path)
623
624     def get_results_tab(self):
625         return self.results_tab
626
627     def get_compare_tab(self):
628         return self.compare_tab
629
630     def get_data(self):
631         return self.data_tab.get_data()
632
633     def get_line(self):
634         return self.line
635
636     def get_n_sd(self):
637         return self.n_sd
638
639     def set_n_sd(self, n):
640         if ',' in n:
641             n = n.replace(',', '.')
642         self.n_sd = float(n)
643         self.need_update = True
644
645     def get_sigma_values(self):
646         return self.sigma_values
647
648     def get_update(self):
649         return self.need_update
650
651     def set_update(self, boolean):
652         self.need_update = boolean
653
654     def get_settings(self):
655         return self.parent.get_settings()
656
657     def get_x_lim(self):
658         return self.parent.get_x_lim()
659
660     def set_x_lim(self, x_lim):
661         self.parent.set_x_lim(x_lim)
662
663     def get_y_lim(self):
664         return self.parent.get_y_lim()
665
666     def set_y_lim(self, y_lim):
667         self.parent.set_y_lim(y_lim)
668
669     def on_error(self, error_message):
670         self.parent.on_error(error_message)
671
672     def save_project(self, path):
673         self.parent.save_project(path)
674
675     def reset(self):
676         self.methods_checked = self.METHODS_CHECKED.copy()
677         self.active_methods = self.ACTIVE_METHODS.copy()
678         self.iso_options = self.ISO_OPTIONS.copy()
679         self.n_sd = self.N_SD
680         self.manual_lines = []
681         self.sigma_values = []
682         self.n_values = []
683         self.data_tab.clear_grid()
684         self.line = Line()
685         self.plot_tab.clear_plots()
686         self.results_tab.clear()
687         self.compare_tab.clear()
688
689
690 # noinspection PyBroadException
691 class DataTab(wx.Panel):
692     def __init__(self, parent):
693         wx.Panel.__init__(self, parent=parent, id=wx.ID_ANY, style=wx.WANTS_CHARS)
694
695         self.parent = parent
696         self.delimiter = self.parent.delimiter
697
698         self.data_changed = False
699
700         default_font = wx.Font(14, wx.DEFAULT, wx.NORMAL, wx.NORMAL)
701
702         hsizer = wx.BoxSizer(wx.HORIZONTAL)
703         self.vsizer = wx.BoxSizer(wx.VERTICAL)
704         vsizer2 = wx.BoxSizer(wx.VERTICAL)
```

```
705
706         self.grid = wx.grid.Grid(self, -1)
707
708         self.grid.SetFont(default_font)
709         self.grid.CreateGrid(12, 2)
710         self.grid.SetColLabelValue(0, u'\N{GREEK CAPITAL LETTER DELTA} \N{GREEK SMALL LETTER SIGMA}')
711         self.grid.SetColLabelValue(1, 'N')
712         self.grid.SetRowLabelSize(40)
713         self.grid.EnableDragColSize(True)
714         self.grid.SetColSize(0, 100)
715         self.grid.SetColSize(1, 150)
716         self.Bind(wx.grid.EVT_GRID_CELL_CHANGE, self.on_change_cell)
717
718         add_row_btn = wx.Button(self, wx.ID_ADD, label='Add rows', size=(100, 40))
719         self.Bind(wx.EVT_BUTTON, self.on_add_row, add_row_btn)
720
721         import_btn = wx.Button(self, wx.ID_OPEN, label='Import data', size=(100, 40))
722         save_btn = wx.Button(self, wx.ID_SAVE, label='Save data', size=(100, 40))
723         self.Bind(wx.EVT_BUTTON, self.on_import, import_btn)
724         self.Bind(wx.EVT_BUTTON, self.on_save, save_btn)
725
726         calculate_btn = wx.Button(self, wx.ID_ANY, label='Calculate', size=(100, 100))
727         self.Bind(wx.EVT_BUTTON, self.on_calculate, calculate_btn)
728
729         self.vsizer.Add(self.grid, 1, wx.EXPAND | wx.ALL)
730         self.vsizer.Add(add_row_btn, 0)
731
732         vsizer2.Add(import_btn, 0)
733         vsizer2.Add(save_btn, 0)
734         vsizer2.Add((-1, -1), -1, wx.EXPAND)
735         vsizer2.Add(calculate_btn, 0)
736
737         hsizer.Add(self.vsizer, -1, wx.EXPAND)
738         hsizer.Add(vsizer2, 0, wx.EXPAND)
739
740         self.SetSizer(hsizer)
741
742     def on_change_cell(self, event=None):
743         self.parent.set_update(True)
744         self.data_changed = True
745
746     def on_add_row(self, event=None):
747         self.grid.AppendRows()
748
749     def on_import(self, event=None):
750         wildcard = "Accepted file formats (*.csv; *.txt)|*.csv;*.txt"
751         dialog = wx.FileDialog(self, 'Import data file', wildcard=wildcard, style=wx.OPEN)
752         if dialog.ShowModal() == wx.ID_OK:
753             path = dialog.GetPath()
754             self.import_data(path)
755             self.parent.set_update(True)
756             self.on_calculate()
757             dialog.Destroy()
758
759     def on_save(self, event=None):
760         wildcard = "Accepted file formats (*.csv; *.txt)|*.csv;*.txt"
761         dialog = wx.FileDialog(self, 'Save as', wildcard=wildcard, style=wx.FD_SAVE)
762         if dialog.ShowModal() == wx.ID_OK:
763             path = dialog.GetPath()
764             self.save_data(path)
765             dialog.Destroy()
766
767     def import_data(self, path):
768         file_name, file_format = os.path.splitext(path)
769         sigma_values = []
770         n_values = []
771         with open(path, 'r') as infile:
772             for data in infile.readlines():
773                 try:
774                     sigma, n = data.split(self.delimiter[file_format])
775                     sigma_values.append(eval(sigma))
776                     n_values.append(eval(n))
777                 except:
778                     pass  # header or text value
779         self.set_data(sigma_values, n_values)
780
781     def set_data(self, sigma_values, n_values):
782         self.grid.ClearGrid()
783         grid_rows = self.grid.GetNumberRows()
784         if grid_rows < len(sigma_values):
785             self.grid.AppendRows(numRows=(len(sigma_values) - grid_rows))
786         for i in range(len(sigma_values)):
787             self.grid.SetCellValue(i, 0, str(sigma_values[i]))
788             self.grid.SetCellValue(i, 1, str(n_values[i]))
789         self.data_changed = True
790
791     def get_data(self):
792         if self.data_changed:
```

```
793             sigma_values = []
794             n_values = []
795             n_too_high = False
796             n_too_low = False
797             rows = self.grid.GetNumberRows()
798             for row in range(rows):
799                 sigma = self.grid.GetCellValue(row, 0)
800                 n = self.grid.GetCellValue(row, 1)
801                 if sigma and n:
802                     if ',' in sigma:
803                         sigma = sigma.replace(',', '.')
804                     if ',' in n:
805                         n = n.replace(',', '.')
806                     good_values = True
807                     try:
808                         sigma = eval(sigma)
809                     except:
810                         self.parent.on_error('Got invalid sigma value: {}'.format(sigma))
811                         good_values = False
812                     try:
813                         n = eval(n)
814                     except:
815                         self.parent.on_error('Got invalid N value: {}'.format(n))
816                         good_values = False
817                     if n > 5*1e6:
818                         n_too_high = True
819                     if n < 1e4:
820                         n_too_low = True
821                     if good_values:
822                         sigma_values.append(sigma)
823                         n_values.append(n)
824             if n_too_high:
825                 warning = 'N values in dataset is higher than 5*10^6 and outside expected values.'
826                 self.parent.on_error(warning)
827             if n_too_low:
828                 warning = 'N values in dataset is lower than 10^4, and is outside expected values.'
829                 self.parent.on_error(warning)
830             self.parent.sigma_values = sigma_values
831             self.parent.n_values = n_values
832 
833             sigma_min = int(np.min(sigma_values))
834             n_dig = len(str(sigma_min))
835             correction = 10.**(n_dig-1)
836             y_min = m.floor(sigma_min/correction)*correction
837             if y_min == sigma_min:
838                 y_min -= correction
839 
840             sigma_max = int(np.max(sigma_values))
841             n_dig = len(str(sigma_max))
842             correction = 10.**(n_dig-1)
843             y_max = m.ceil(sigma_max/correction)*correction
844             if y_max == sigma_max:
845                 y_max += correction
846             self.parent.set_y_lim((y_min, y_max))
847             self.data_changed = False
848         return self.parent.sigma_values, self.parent.n_values
849 
850     def save_data(self, path):
851         if '.txt' in path:
852             file_format = '.txt'
853         elif '.csv' in path:
854             file_format = '.csv'
855         else:
856             file_format = '.txt'
857             path += file_format
858         sigma_values, n_values = self.get_data()
859         header = 'Sigma' + self.delimiter[file_format] + 'N'
860 
861         with open(path, 'w') as infile:
862             infile.write(header)
863             for sigma, N in zip(sigma_values, n_values):
864                 infile.write('\n' + str(sigma) + self.delimiter[file_format] + str(N))
865 
866     def on_calculate(self, event=None):
867         self.parent.get_results_tab().on_calculate()
868         self.parent.get_plot_tab().on_plot()
869         self.parent.get_compare_tab().update_line_value()
870 
871     def clear_grid(self):
872         self.grid.ClearGrid()
873 
874 
875 class PlotTab(wx.Panel):
876     def __init__(self, parent):
877         super(PlotTab, self).__init__(parent=parent, id=wx.ID_ANY)
878         self.notebook = PlotNotebookStructure(parent, self)
879         self.sn_tab = self.notebook.get_sn_tab()
880         self.residual_tab = self.notebook.get_residual_tab()
```

```
881         self.norm_prob_tab = self.notebook.get_norm_prob_tab()
882
883         vsizer = wx.BoxSizer(wx.VERTICAL)
884         vsizer.Add(self.notebook, 1, wx.ALL | wx.EXPAND, 5)
885
886         self.SetSizer(vsizer)
887         self.Show()
888
889     def get_sn_tab(self):
890         return self.sn_tab
891
892     def on_plot(self):
893         self.sn_tab.on_plot()
894         self.residual_tab.on_plot()
895         self.norm_prob_tab.on_plot()
896
897     def update_checkbox(self):
898         self.sn_tab.update_checkbox()
899
900     def update_radio_btn(self):
901         self.sn_tab.update_radio_btn()
902
903     def save_plots(self, folder_path):
904         self.sn_tab.save_plot(folder_path)
905         self.residual_tab.save_plot(folder_path)
906         self.norm_prob_tab.save_plot(folder_path)
907
908     def clear_plots(self):
909         self.sn_tab.clear()
910         self.residual_tab.clear()
911         self.norm_prob_tab.clear()
912
913
914 class PlotNotebookStructure(wx.Notebook):
915     def __init__(self, parent, panel):
916         wx.Notebook.__init__(self, panel, id=wx.ID_ANY, style=wx.BK_DEFAULT)
917         self.parent = parent
918
919         self.sn_tab = SNTab(self)
920         self.residual_tab = ResidualTab(self)
921         self.norm_prob_tab = NormProbTab(self)
922         self.AddPage(self.sn_tab, 'S-N plot')
923         self.AddPage(self.residual_tab, 'Residual plot')
924         self.AddPage(self.norm_prob_tab, 'Normal probability plot')
925
926     def on_plot(self):
927         self.sn_tab.on_plot()
928
929     def on_check(self, event=None):
930         self.sn_tab.on_check()
931
932     def update_checkbox(self):
933         self.sn_tab.update_checkbox()
934
935     def get_manual_lines(self):
936         return self.parent.get_manual_lines()
937
938     def set_manual_lines(self, lines):
939         self.parent.set_manual_lines(lines)
940
941     def set_use_manual_lines(self, boolean):
942         self.parent.set_use_manual_lines(boolean)
943
944     def get_use_manual_lines(self):
945         return self.parent.get_use_manual_lines()
946
947     def get_sn_tab(self):
948         return self.sn_tab
949
950     def get_residual_tab(self):
951         return self.residual_tab
952
953     def get_norm_prob_tab(self):
954         return self.norm_prob_tab
955
956     def get_data(self):
957         return self.parent.get_data()
958
959     def get_sigma_values(self):
960         return self.parent.get_sigma_values()
961
962     def get_update(self):
963         return self.parent.get_update()
964
965     def set_update(self, boolean):
966         self.parent.set_update(boolean)
967
968     def get_settings(self):
```

```
969         return self.parent.get_settings()
970
971     def get_methods_checked(self):
972         return self.parent.methods_checked
973
974     def get_iso_options(self):
975         return self.parent.iso_options
976
977     def get_iso_conf(self):
978         return self.parent.get_iso_conf()
979
980     def get_iso_prob(self):
981         return self.parent.get_iso_prob()
982
983     def get_results_tab(self):
984         return self.parent.get_results_tab()
985
986     def get_active_methods(self):
987         return self.parent.active_methods
988
989     def get_line(self):
990         return self.parent.get_line()
991
992     def get_n_sd(self):
993         return self.parent.get_n_sd()
994
995     def set_n_sd(self, n):
996         self.parent.set_n_sd(n)
997
998     def get_x_lim(self):
999         return self.parent.get_x_lim()
1000
1001    def set_x_lim(self, x_lim):
1002        self.parent.set_x_lim(x_lim)
1003
1004    def get_y_lim(self):
1005        return self.parent.get_y_lim()
1006
1007    def set_y_lim(self, y_lim):
1008        self.parent.set_y_lim(y_lim)
1009
1010    def on_error(self, error_message):
1011        self.parent.on_error(error_message)
1012
1013
1014 # noinspection PyShadowingNames
1015 class SNTab(wx.Panel):
1016     def __init__(self, parent):
1017         wx.Panel.__init__(self, parent=parent, id=wx.ID_ANY)
1018         self.parent = parent
1019         self.figure = Figure((10, 10), dpi=72)
1020         self.canvas = FigureCanvas(self, -1, self.figure)
1021
1022         self.plot_styles = ['-', '--', ':', '-.', 'v-', 'v--', 'v:', 'v-.', '^-', '^--', '^:', '^-.']
1023         font = wx.Font(12, wx.DECORATIVE, wx.NORMAL, wx.BOLD)
1024         self.canvas.SetFont(font)
1025
1026         self.n_sd_changed = False
1027         self.checked_changed = False
1028
1029         self.grid = True
1030         self.x_ticks = False
1031         self.y_ticks = True
1032
1033         self.axes = self.figure.add_subplot(111)
1034         self.axes.set_yscale('log')
1035         self.axes.set_xscale('log')
1036         self.axes.set_xlim(self.parent.get_x_lim())
1037         self.axes.set_ylim(self.parent.get_y_lim())
1038         self.toolbar = NavigationToolbar(self.canvas)
1039         settings_text = wx.StaticText(self, -1, 'Show:')
1040         settings_text.SetFont(font)
1041         self.show_points_cb = wx.CheckBox(self, label='Data points')
1042         self.reg_line_cb = wx.CheckBox(self, label='Regression line')
1043         self.conf_line_cb = wx.CheckBox(self, label='Confidence interval')
1044         design_text = wx.StaticText(self, -1, 'Design lines:')
1045         self.design_2_sd_cb = wx.CheckBox(self, label='2 standard deviations (SD)')
1046         self.design_iiw_cb = wx.CheckBox(self, label='IIW')
1047         self.design_dnv_known_cb = wx.CheckBox(self, label='DNV, SD known')
1048         self.design_dnv_unknown_cb = wx.CheckBox(self, label='DNV, SD unknown')
1049
1050         self.design_iso_cb = wx.CheckBox(self, label='ISO (simplified)')
1051         hsizer_confidence_txt = wx.BoxSizer(wx.HORIZONTAL)
1052         iso_text_conf = wx.StaticText(self, -1, 'Confidence level:')
1053         hsizer_confidence_txt.Add(iso_text_conf, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
1054         hsizer_confidence = wx.BoxSizer(wx.HORIZONTAL)
1055         self.conf_90 = wx.RadioButton(self, label='90%', style=wx.RB_GROUP)
1056         self.conf_95 = wx.RadioButton(self, label='95%')
```

```
        hsizer_confidence.Add(self.conf_90, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
        hsizer_confidence.Add(self.conf_95, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
        hsizer_probability_txt = wx.BoxSizer(wx.HORIZONTAL)
        iso_text_prob = wx.StaticText(self, -1, 'Probability:')
        hsizer_probability_txt.Add(iso_text_prob, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
        hsizer_probability = wx.BoxSizer(wx.HORIZONTAL)
        self.prob_10 = wx.RadioButton(self, label='10%', style=wx.RB_GROUP)
        self.prob_05 = wx.RadioButton(self, label='5%')
        self.prob_01 = wx.RadioButton(self, label='1%')
        hsizer_probability.Add(self.prob_10, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
        hsizer_probability.Add(self.prob_05, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
        hsizer_probability.Add(self.prob_01, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)

        self.design_n_sd_cb = wx.CheckBox(self, label='n SD')
        self.n_sd = wx.TextCtrl(self, value=str(self.parent.get_n_sd()))
        self.n_sd.Disable()
        self.show_points_cb.SetValue(True)
        self.update_checkbox()
        self.update_radio_btn()
        self.Bind(wx.EVT_CHECKBOX, self.on_check)
        self.Bind(wx.EVT_RADIOBUTTON, self.on_radio_btn)

        self.extra_cb = wx.CheckBox(self, label='Manual line(s)')
        self.extra_cb.SetValue(False)
        self.extra_cb.Show(False)

        line_btn = wx.Button(self, -1, 'Add/edit manual lines', size=(100, 40))
        self.Bind(wx.EVT_BUTTON, self.on_new_line, line_btn)

        axes_btn = wx.Button(self, -1, 'Adjust plot', size=(100, 40))
        self.Bind(wx.EVT_BUTTON, self.on_axes, axes_btn)

        plot_btn = wx.Button(self, -1, 'Show', size=(100, 100))
        self.Bind(wx.EVT_BUTTON, self.on_plot, plot_btn)

        hsizer = wx.BoxSizer(wx.HORIZONTAL)
        vsizer = wx.BoxSizer(wx.VERTICAL)
        vsizer2 = wx.BoxSizer(wx.VERTICAL)

        vsizer.Add(self.canvas, 1, wx.LEFT | wx.TOP | wx.BOTTOM | wx.GROW)
        vsizer.Add(self.toolbar, 0, wx.EXPAND)

        vsizer2.Add(settings_text, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
        vsizer2.Add(self.show_points_cb, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
        vsizer2.Add(self.reg_line_cb, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
        vsizer2.Add(self.conf_line_cb, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)

        vsizer2.Add(design_text, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
        vsizer2.Add(self.design_2_sd_cb, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
        vsizer2.Add(self.design_iiw_cb, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
        vsizer2.Add(self.design_dnv_known_cb, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
        vsizer2.Add(self.design_dnv_unknown_cb, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
        vsizer2.Add(self.design_iso_cb, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
        vsizer2.Add(hsizer_confidence_txt, 0, flag=wx.EXPAND | wx.LEFT, border=20)
        vsizer2.Add(hsizer_confidence, 0, flag=wx.EXPAND | wx.LEFT, border=20)
        vsizer2.Add(hsizer_probability_txt, 0, flag=wx.EXPAND | wx.LEFT, border=20)
        vsizer2.Add(hsizer_probability, 0, flag=wx.EXPAND | wx.LEFT, border=20)
        vsizer2.Add(self.design_n_sd_cb, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
        vsizer2.Add(self.n_sd, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
        vsizer2.Add(self.extra_cb, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
        vsizer2.Add((-1, -1), -1, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
        vsizer2.Add(line_btn, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
        vsizer2.Add(axes_btn, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
        vsizer2.Add(plot_btn, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)

        hsizer.Add(vsizer, -1, wx.LEFT | wx.TOP | wx.BOTTOM | wx.GROW)
        hsizer.Add(vsizer2, 0, wx.LEFT | wx.TOP | wx.BOTTOM | wx.GROW)

        self.Bind(wx.EVT_TEXT, self.on_change_sd)

        self.SetSizer(hsizer)

    def get_axes_settings(self):
        if self.x_ticks:
            minor_x = True
        else:
            minor_x = False
        if self.y_ticks:
            minor_y = True
        else:
            minor_y = False
        x_lim = self.parent.get_x_lim()
        y_lim = self.parent.get_y_lim()
        return {'show_grid': self.grid, 'x_lim': x_lim, 'show_minor_x': minor_x,
                'y_lim': y_lim, 'show_minor_y': minor_y}

    def set_axes_settings(self, settings_dict):
        self.grid = settings_dict['show_grid']
```

```
        self.parent.set_x_lim(settings_dict['x_lim'])
        self.parent.set_y_lim(settings_dict['y_lim'])

        if settings_dict['show_minor_x']:
            self.set_xticks(True)
        else:
            self.set_xticks(False)
        if settings_dict['show_minor_y']:
            self.set_yticks(True)
        else:
            self.set_xticks(False)

    def get_manual_lines(self):
        return self.parent.get_manual_lines()

    def set_manual_lines(self, lines):
        self.parent.set_manual_lines(lines)
        self.extra_cb.SetValue(True)
        self.extra_cb.Show(True)
        self.parent.set_use_manual_lines(True)
        results_tab = self.parent.get_results_tab()
        results_tab.extra_cb.SetValue(True)
        results_tab.extra_cb.Show(True)
        self.GetSizer().Layout()

    def on_check(self, event=None):
        methods_checked = self.parent.get_methods_checked()
        active_methods = self.parent.get_active_methods()
        for box, _ in methods_checked.items():
            value = getattr(self, box).GetValue()
            methods_checked[box] = value
            active_methods[box[:-3]] = value
        if self.design_n_sd_cb.GetValue():
            self.n_sd.Enable()
        else:
            self.n_sd.Disable()
        if self.parent.get_manual_lines() and self.extra_cb.GetValue():
            self.parent.set_use_manual_lines(True)
        else:
            self.parent.set_use_manual_lines(False)
        self.checked_changed = True
        results_tab = self.parent.get_results_tab()
        results_tab.update_checkbox()

    def update_checkbox(self):
        for box, value in self.parent.get_methods_checked().items():
            getattr(self, box).SetValue(value)
        if self.design_n_sd_cb.GetValue():
            self.n_sd.Enable()
            self.n_sd.SetValue(str(self.parent.get_n_sd()))
            self.n_sd_changed = False
        else:
            self.n_sd.SetValue(str(self.parent.get_n_sd()))
            self.n_sd.Disable()
        if self.parent.get_use_manual_lines() and self.parent.get_manual_lines():
            self.extra_cb.SetValue(True)
        elif self.parent.get_manual_lines():
            self.extra_cb.SetValue(False)

    def on_radio_btn(self, event=None):
        iso_options = self.parent.get_iso_options()
        for btn, _ in iso_options.items():
            value = getattr(self, btn).GetValue()
            iso_options[btn] = value
        results_tab = self.parent.get_results_tab()
        results_tab.update_radio_btn()
        self.checked_changed = True

    def update_radio_btn(self):
        for btn, value in self.parent.get_iso_options().items():
            getattr(self, btn).SetValue(value)

    def on_change_sd(self, event=None):
        self.n_sd_changed = True

    def on_plot(self, event=None):
        line = self.parent.get_line()
        sigma_values, n_values = self.parent.get_data()
        results_tab = self.parent.get_results_tab()
        settings = self.parent.get_settings()
        main_label = settings['line_name']
        if ' ' in main_label:
            main_label = main_label.replace(' ', '\ ')
        if '_' in main_label:
            main_label = main_label.replace('_', '\_')
        extra_design_label = settings['use_line_name']

        if self.n_sd_changed:
```

```
            self.parent.set_n_sd(self.n_sd.GetValue())
            self.parent.get_results_tab().update_checkbox()
            self.n_sd_changed = False
        if self.parent.get_update():
            self.n_sd.SetValue(str(self.parent.get_n_sd()))
            results_tab.on_calculate()
        self.axes.clear()
        legend = []
        if sigma_values:
            show_points = self.show_points_cb.GetValue()
            if show_points:
                self.axes.plot(n_values, sigma_values, 'bo')
                legend.append('Data')
            if self.reg_line_cb.GetValue():
                n, sigma, label, name = line.get_line_points('reg_line')
                label = label % main_label
                self.axes.plot(n, sigma, 'b-')
                legend.append(label)
            if self.conf_line_cb.GetValue():
                n1, n2, sigma = line.get_ci_points()
                self.axes.plot(n1+n2, sigma+sigma, 'b:')
                legend.append('Confidence interval')
            n_design = 0
            for method, use in self.parent.get_active_methods().items():
                if use and method != 'reg_line':
                    if 'n_sd' in method:
                        n_sd = self.parent.get_n_sd()
                        n, sigma, label, name = line.get_line_points(method, n_sd)
                    elif 'iso' in method:
                        conf = self.parent.get_iso_conf()
                        prob = self.parent.get_iso_prob()
                        name = 'iso_'+conf+'_'+prob
                        n, sigma, label, name = line.get_line_points(name)
                    else:
                        n, sigma, label, name = line.get_line_points(method)
                    style = 'g' + self.plot_styles[n_design]
                    n_design += 1
                    self.axes.plot(n, sigma, style)
                    if extra_design_label:
                        name = main_label+'\ '+name
                    label = label % name
                    legend.append(label)

        if self.extra_cb.GetValue():
            n_manual = 0
            for manual_line in self.parent.get_manual_lines():
                n_data, sigma_data, label, name = manual_line.get_line_points()
                manual_style = 'r' + self.plot_styles[n_manual]
                self.axes.plot(n_data, sigma_data, manual_style)
                if ' ' in name:
                    name = name.replace(' ', '\ ')
                if '_' in name:
                    name = name.replace('_', '\_')
                legend.append(label % name)
                n_manual += 1

        self.axes.legend(legend)

        self.axes.set_yscale('log')
        self.axes.set_xscale('log')
        self.axes.set_xlim(self.parent.get_x_lim())
        self.axes.set_ylim(self.parent.get_y_lim())
        if self.y_ticks:
            self.axes.yaxis.set_minor_formatter(FormatStrFormatter("%d"))
        if self.x_ticks:
            self.axes.xaxis.set_minor_formatter(FormatStrFormatter("%d"))
        if self.grid:
            self.axes.grid(True, which='major', linestyle='-')
            self.axes.grid(True, which='minor', linestyle='--')
        else:
            self.axes.grid(False, which='major')
            self.axes.grid(False, which='minor')

        font = {'weight': 'normal',
                'size': 18}

        self.axes.set_xlabel(r'Number of Cycles [N]', fontdict=font)
        self.axes.set_ylabel(r'Stress Range, $\Delta \sigma $ [MPa]', fontdict=font)

        self.canvas.draw()

        if self.checked_changed:
            results_tab = self.parent.get_results_tab()
            results_tab.on_calculate()
            self.checked_changed = False

    def on_new_line(self, event=None):
        NewLineWindow(self)
```

```
1321
1322    def on_axes(self, event=None):
1323        PlotWindow(self)
1324
1325    def get_x_lim(self):
1326        return self.parent.get_x_lim()
1327
1328    def get_y_lim(self):
1329        return self.parent.get_y_lim()
1330
1331    def set_x_lim(self, x_lim):
1332        self.parent.set_x_lim(x_lim)
1333
1334    def set_y_lim(self, y_lim):
1335        self.parent.set_y_lim(y_lim)
1336
1337    def set_xticks(self, x_ticks):
1338        self.x_ticks = x_ticks
1339
1340    def set_yticks(self, y_ticks):
1341        self.y_ticks = y_ticks
1342
1343    def get_xticks(self):
1344        return self.x_ticks
1345
1346    def get_yticks(self):
1347        return self.y_ticks
1348
1349    def set_show_grid(self, boolean):
1350        self.grid = boolean
1351
1352    def get_show_grid(self):
1353        return self.grid
1354
1355    def get_sigma_values(self):
1356        return self.parent.get_sigma_values()
1357
1358    def save_plot(self, folder_path):
1359        self.on_plot()
1360        name = folder_path + '\\' + 'S-N_curve'
1361
1362        self.figure.savefig(name, dpi=float(self.parent.get_settings()['dpi']))
1363        self.on_plot()  # Reset canvas
1364
1365    def clear(self):
1366        self.update_checkbox()
1367        self.update_radio_btn()
1368        self.axes.clear()
1369        self.canvas.draw()
1370        self.extra_cb.SetValue(False)
1371        self.extra_cb.Show(False)
1372        self.parent.set_use_manual_lines(False)
1373        results_tab = self.parent.get_results_tab()
1374        results_tab.extra_cb.SetValue(False)
1375        results_tab.extra_cb.Show(False)
1376
1377
1378  # noinspection PyPep8Naming
1379  class ResidualTab(wx.Panel):
1380      def __init__(self, parent):
1381          wx.Panel.__init__(self, parent=parent, id=wx.ID_ANY)
1382          self.parent = parent
1383          self.figure = Figure((10, 10), dpi=72)
1384          self.canvas = FigureCanvas(self, -1, self.figure)
1385          font = wx.Font(12, wx.DECORATIVE, wx.NORMAL, wx.BOLD)
1386          self.canvas.SetFont(font)
1387
1388          self.x_ticks = False
1389          self.y_ticks = False
1390
1391          self.axes = self.figure.add_subplot(111)
1392          self.toolbar = NavigationToolbar(self.canvas)
1393
1394          vsizer = wx.BoxSizer(wx.VERTICAL)
1395
1396          vsizer.Add(self.canvas, 1, wx.LEFT | wx.TOP | wx.BOTTOM | wx.GROW)
1397          vsizer.Add(self.toolbar, 0, wx.EXPAND)
1398
1399          self.SetSizer(vsizer)
1400
1401      def on_plot(self, event=None):
1402          line = self.parent.get_line()
1403          sigma_values, n_values = self.parent.get_data()
1404          self.axes.clear()
1405
1406          if sigma_values:
1407              residuals = line.get_standarized_residuals()
1408              y_values = line.get_estimated_y()
```

```
            N = [10**y for y in y_values]

            av_residual = sum(residuals)/len(residuals)

            self.axes.plot(N, residuals, 'bo')
            zero_line = [av_residual for _ in N]
            self.axes.plot(N, zero_line, 'b-')

        font = {'weight': 'normal',
                'size': 18}
        self.axes.set_ylabel(r'Standarized residuals', fontdict=font)
        self.axes.set_xlabel(r'Predicted life', fontdict=font)
        self.axes.set_xscale('log')
        self.canvas.draw()

    def save_plot(self, folder_path):
        self.on_plot()
        name = folder_path + '\\' + 'residual_plot'

        self.figure.savefig(name, dpi=float(self.parent.get_settings()['dpi']))
        self.on_plot()  # Reset canvas

    def clear(self):
        self.axes.clear()


class NormProbTab(wx.Panel):
    def __init__(self, parent):
        wx.Panel.__init__(self, parent=parent, id=wx.ID_ANY)
        self.parent = parent
        self.figure = Figure((10, 10), dpi=72)
        self.canvas = FigureCanvas(self, -1, self.figure)
        font = wx.Font(12, wx.DECORATIVE, wx.NORMAL, wx.BOLD)
        self.canvas.SetFont(font)

        self.x_ticks = False
        self.y_ticks = False

        self.axes = self.figure.add_subplot(111)
        self.toolbar = NavigationToolbar(self.canvas)

        vsizer = wx.BoxSizer(wx.VERTICAL)

        vsizer.Add(self.canvas, 1, wx.LEFT | wx.TOP | wx.BOTTOM | wx.GROW)
        vsizer.Add(self.toolbar, 0, wx.EXPAND)

        self.SetSizer(vsizer)

    def on_plot(self, event=None):
        line = self.parent.get_line()
        sigma_values, n_values = self.parent.get_data()
        self.axes.clear()
        if sigma_values:
            residuals = line.get_standarized_residuals()
            residuals_sorted = sorted(residuals)
            points = line.get_norm_points()
            self.axes.plot(points, residuals_sorted, 'bo')
        font = {'weight': 'normal',
                'size': 18}
        self.axes.set_ylabel(r'Standarized residuals', fontdict=font)
        self.axes.set_xlabel(r'Normal probability', fontdict=font)
        self.canvas.draw()

    def save_plot(self, folder_path):
        self.on_plot()
        name = folder_path + '\\' + 'norm_prob_plot'

        self.figure.savefig(name, dpi=float(self.parent.get_settings()['dpi']))
        self.on_plot()  # Reset canvas

    def clear(self):
        self.axes.clear()


class ResultsTab(wx.Panel):
    def __init__(self, parent):
        wx.Panel.__init__(self, parent=parent, id=wx.ID_ANY)
        self.parent = parent
        font = wx.Font(12, wx.DECORATIVE, wx.NORMAL, wx.BOLD)

        self.n_sd_changed = False
        self.checked_changed = False

        self.figure = Figure((10, 10), dpi=72)
        self.canvas = FigureCanvas(self, -1, self.figure)
        self.axes = self.figure.add_subplot(111)
        self.toolbar = NavigationToolbar(self.canvas)
        self.figure.set_facecolor('#FFFFFF')
```

```
1497
1498        report_text = wx.StaticText(self, -1, 'Report:')
1499        report_text.SetFont(font)
1500        self.reg_line_cb = wx.CheckBox(self, label='Regression line')
1501        design_text = wx.StaticText(self, -1, 'Design lines:')
1502        self.design_2_sd_cb = wx.CheckBox(self, label='2 standard deviations (SD)')
1503        self.design_iiw_cb = wx.CheckBox(self, label='IIW')
1504        self.design_dnv_known_cb = wx.CheckBox(self, label='DNV, SD known')
1505        self.design_dnv_unknown_cb = wx.CheckBox(self, label='DNV, SD unknown')
1506
1507        self.design_iso_cb = wx.CheckBox(self, label='ISO (simplified)')
1508        hsizer_confidence_txt = wx.BoxSizer(wx.HORIZONTAL)
1509        iso_text_conf = wx.StaticText(self, -1, 'Confidence level:')
1510        hsizer_confidence_txt.Add(iso_text_conf, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
1511        hsizer_confidence = wx.BoxSizer(wx.HORIZONTAL)
1512        self.conf_90 = wx.RadioButton(self, label='90%', style=wx.RB_GROUP)
1513        self.conf_95 = wx.RadioButton(self, label='95%')
1514        hsizer_confidence.Add(self.conf_90, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
1515        hsizer_confidence.Add(self.conf_95, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
1516        hsizer_probability_txt = wx.BoxSizer(wx.HORIZONTAL)
1517        iso_text_prob = wx.StaticText(self, -1, 'Probability:')
1518        hsizer_probability_txt.Add(iso_text_prob, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
1519        hsizer_probability = wx.BoxSizer(wx.HORIZONTAL)
1520        self.prob_10 = wx.RadioButton(self, label='10%', style=wx.RB_GROUP)
1521        self.prob_05 = wx.RadioButton(self, label='5%')
1522        self.prob_01 = wx.RadioButton(self, label='1%')
1523        hsizer_probability.Add(self.prob_10, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
1524        hsizer_probability.Add(self.prob_05, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
1525        hsizer_probability.Add(self.prob_01, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
1526
1527        self.design_n_sd_cb = wx.CheckBox(self, label='n SD')
1528        self.n_sd = wx.TextCtrl(self, value=str(self.parent.get_n_sd()))
1529        self.n_sd.Disable()
1530
1531        self.extra_cb = wx.CheckBox(self, label='Manual line(s)')
1532        self.extra_cb.SetValue(False)
1533        self.extra_cb.Show(False)
1534
1535        self.update_checkbox()
1536        self.update_radio_btn()
1537        self.Bind(wx.EVT_CHECKBOX, self.on_check)
1538        self.Bind(wx.EVT_RADIOBUTTON, self.on_radio_btn)
1539
1540        report_btn = wx.Button(self, -1, 'Create report', size=(100, 60))
1541        self.Bind(wx.EVT_BUTTON, self.on_report, report_btn)
1542        calculate_btn = wx.Button(self, -1, 'Show', size=(100, 60))
1543        self.Bind(wx.EVT_BUTTON, self.on_calculate, calculate_btn)
1544
1545        hsizer = wx.BoxSizer(wx.HORIZONTAL)
1546        vsizer = wx.BoxSizer(wx.VERTICAL)
1547        vsizer2 = wx.BoxSizer(wx.VERTICAL)
1548
1549        vsizer.Add(report_text, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
1550        vsizer.Add(self.reg_line_cb, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
1551
1552        vsizer.Add(design_text, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
1553        vsizer.Add(self.design_2_sd_cb, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
1554        vsizer.Add(self.design_iiw_cb, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
1555        vsizer.Add(self.design_dnv_known_cb, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
1556        vsizer.Add(self.design_dnv_unknown_cb, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
1557
1558        vsizer.Add(self.design_iso_cb, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
1559        vsizer.Add(hsizer_confidence_txt, 0, flag=wx.EXPAND | wx.LEFT, border=20)
1560        vsizer.Add(hsizer_confidence, 0, flag=wx.EXPAND | wx.LEFT, border=20)
1561        vsizer.Add(hsizer_probability_txt, 0, flag=wx.EXPAND | wx.LEFT, border=20)
1562        vsizer.Add(hsizer_probability, 0, flag=wx.EXPAND | wx.LEFT, border=20)
1563
1564        vsizer.Add(self.design_n_sd_cb, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
1565        vsizer.Add(self.n_sd, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
1566        vsizer.Add(self.extra_cb, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
1567
1568        vsizer.Add(calculate_btn, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
1569        vsizer.Add(report_btn, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
1570
1571        results_text = wx.StaticText(self, -1, 'Results:')
1572        results_text.SetFont(font)
1573        self.results_print = wx.StaticText(self, -1, '')
1574        vsizer2.Add(results_text, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
1575        vsizer2.Add(self.results_print, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
1576        vsizer2.Add(self.canvas, 1, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
1577        vsizer2.Add(self.toolbar, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
1578
1579        self.Bind(wx.EVT_TEXT, self.on_change_sd)
1580
1581        hsizer.Add(vsizer, 0)
1582        hsizer.Add(vsizer2, -1)
1583        self.SetSizer(hsizer)
1584
```

```
1585    def on_check(self, event=None):
1586        for box, _ in self.parent.methods_checked.items():
1587            value = getattr(self, box).GetValue()
1588            self.parent.methods_checked[box] = value
1589            self.parent.active_methods[box[:-3]] = value
1590        if self.design_n_sd_cb.GetValue():
1591            self.n_sd.Enable()
1592        else:
1593            self.n_sd.Disable()
1594        if self.parent.get_manual_lines() and self.extra_cb.GetValue():
1595            self.parent.set_use_manual_lines(True)
1596        else:
1597            self.parent.set_use_manual_lines(False)
1598        self.checked_changed = True
1599        plot_tab = self.parent.get_plot_tab()
1600        plot_tab.update_checkbox()
1601
1602    def update_checkbox(self):
1603        for box, value in self.parent.methods_checked.items():
1604            getattr(self, box).SetValue(value)
1605        if self.design_n_sd_cb.GetValue():
1606            self.n_sd.Enable()
1607            self.n_sd.SetValue(str(self.parent.get_n_sd()))
1608            self.n_sd_changed = False
1609        else:
1610            self.n_sd.SetValue(str(self.parent.get_n_sd()))
1611            self.n_sd.Disable()
1612        if self.parent.get_use_manual_lines() and self.parent.get_manual_lines():
1613            self.extra_cb.SetValue(True)
1614        elif self.parent.get_manual_lines():
1615            self.extra_cb.SetValue(False)
1616
1617    def on_radio_btn(self, event=None):
1618        iso_options = self.parent.get_iso_options()
1619        for btn, _ in iso_options.items():
1620            value = getattr(self, btn).GetValue()
1621            iso_options[btn] = value
1622        plot_tab = self.parent.get_plot_tab()
1623        plot_tab.update_radio_btn()
1624        self.checked_changed = True
1625
1626    def update_radio_btn(self):
1627        for btn, value in self.parent.get_iso_options().items():
1628            getattr(self, btn).SetValue(value)
1629
1630    def on_change_sd(self, event=None):
1631        self.n_sd_changed = True
1632
1633    def set_n_sd(self, n):
1634        self.n_sd.SetValue(n)
1635
1636    def on_report(self, event=None):
1637        sigma_values, n_values = self.parent.get_data()
1638        if sigma_values:
1639            dialog = wx.DirDialog(self, 'Choose a location for the report folder', style=wx.SAVE)
1640            if dialog.ShowModal() == wx.ID_OK:
1641                path = dialog.GetPath()
1642                self.report(path)
1643                dialog.Destroy()
1644        else:
1645            PopupWindow(self, 'Reporting terminated', 'No data detected', size=(300, 200))
1646
1647    def on_calculate(self, event=None):
1648        if self.n_sd_changed:
1649            self.parent.set_n_sd(self.n_sd.GetValue())
1650            self.parent.get_plot_tab().update_checkbox()
1651            self.n_sd_changed = False
1652        if self.parent.get_update():
1653            self.n_sd.SetValue(str(self.parent.get_n_sd()))
1654            sigma_values, n_values = self.parent.get_data()
1655
1656            if not sigma_values:
1657                return
1658            self.parent.get_line().update_values(n_values, sigma_values)
1659            self.parent.set_update(False)
1660            self.show_results()
1661            self.update_ellipse()
1662            self.Layout()
1663        if self.checked_changed:
1664            self.checked_changed = False
1665            self.parent.get_plot_tab().on_plot()
1666
1667    def show_results(self):
1668        settings = self.parent.get_settings()
1669        main_label = settings['line_name']
1670        if ' ' in main_label:
1671            main_label = main_label.replace(' ', '\ ')
1672        if '_' in main_label:
```

```
            main_label = main_label.replace('_', '\_')
        extra_design_label = settings['use_line_name']

        result_string = ''
        if self.parent.active_methods['reg_line']:
            line, name = self.parent.get_line().get_line_info_wx('reg_line')
            result_string += line % main_label
            result_string += '\n'
        for method, use in self.parent.active_methods.items():
            if method != 'reg_line' and use:
                if 'n_sd' in method:
                    n_sd = self.parent.get_n_sd()
                    line, name = self.parent.get_line().get_line_info_wx(method, n_sd)
                elif 'iso' in method:
                        conf = self.parent.get_iso_conf()
                        prob = self.parent.get_iso_prob()
                        method = 'iso_'+conf+'_'+prob
                        line, name = self.parent.get_line().get_line_info_wx(method)
                else:
                    line, name = self.parent.get_line().get_line_info_wx(method)
                if extra_design_label:
                    name = main_label+'_'+name
                result_string += line % name
                result_string += '\n'
        if self.extra_cb.GetValue():
            for line in self.parent.get_manual_lines():
                line, name = line.get_line_info_wx()
                result_string += line % name
                result_string += '\n'
        result_string += '\n'
        result_string += self.parent.get_line().get_results_for_print()

        self.results_print.SetLabel(result_string)

    def update_ellipse(self):
        line = self.parent.get_line()
        sigma_values, n_values = self.parent.get_data()

        if sigma_values:
            self.axes.clear()
            c_values, m_values = line.get_ellipse_points()
            self.axes.plot(c_values, m_values, 'b-')
            self.axes.fill(c_values, m_values, 'b')

            lines = line.get_CI_lines()
            for c, m in lines:
                self.axes.plot(c, m, 'b--')

            self.axes.set_xscale('log')
            self.axes.set_xlim(min(c_values)/1.5, max(c_values)*1.3)
            self.axes.set_ylim(min(m_values)-0.2, max(m_values)+0.2)

            self.axes.set_xlabel('c-values')
            self.axes.set_ylabel('m-values')
            self.axes.set_title('Simultaneous CI for c and m')
        self.canvas.draw()
        self.Layout()
        self.figure.tight_layout()

    # noinspection PyBroadException
    def report(self, path):
        settings = self.parent.get_settings()
        if self.parent.get_update():
            self.on_calculate()
        foldername = settings['project_name']+'_report'
        folder_num = 1

        for folder in [folder for folder in os.listdir(path) if os.path.isdir(os.path.join(path, folder))]:
            if foldername in folder and '_' in folder:
                elements = folder.split('_')
                try:
                    number = int(elements[-1])
                    if folder_num <= number:
                        folder_num = number + 1
                except:
                    pass
        folder_path = path + '\\' + foldername + '_' + str(folder_num)
        os.makedirs(folder_path)

        self.save_plots(folder_path)

        self.parent.save_project(folder_path+'\\'+settings['project_name']+'_settings.txt')
        if settings['report_format'] == '.pdf':
            self.report_to_pdf(folder_path)
        else:
            self.report_to_txt(folder_path)
        PopupWindow(self, '', 'Report done', size=(300, 200))

```

```
1761    def save_plots(self, folder_path):
1762        self.parent.save_plots(folder_path)
1763        self.save_ellipse(folder_path)
1764
1765    def save_ellipse(self, folder_path):
1766        self.update_ellipse()
1767        name = folder_path + '\\' + 'simultaneous_CI_plot'
1768
1769        self.figure.savefig(name, dpi=float(self.parent.get_settings()['dpi']))
1770        self.update_ellipse()  # Reset canvas
1771
1772    def report_to_pdf(self, folder_path):
1773        sigma_values, n_values = self.parent.data_tab.get_data()
1774        line = self.parent.get_line()
1775        settings = self.parent.get_settings()
1776
1777        design_lines = []
1778        for line_name, active in self.parent.active_methods.iteritems():
1779            if active and line_name != 'reg_line':
1780                if 'iso' in line_name:
1781                    conf = self.parent.get_iso_conf()
1782                    prob = self.parent.get_iso_prob()
1783                    name = 'iso_'+conf+'_'+prob
1784                    design_lines.append(name)
1785                else:
1786                    design_lines.append(line_name)
1787
1788        if self.parent.get_use_manual_lines():
1789            manual_lines = self.parent.get_manual_lines()
1790        else:
1791            manual_lines = []
1792        report_name = self.parent.get_settings()['project_name']+'_report'
1793
1794        use_summary = settings['use_summary']
1795        report = ReportGenerator(folder_path, report_name)
1796        report.make_template(len(sigma_values), len(design_lines), len(manual_lines), use_summary)
1797
1798        content_dict = {'title': settings['title'],
1799                        'author': settings['author'],
1800                        'intro': settings['intro'],
1801                        'sn_plot': 'S-N_curve.png',
1802                        'residual_plot': 'residual_plot.png',
1803                        'norm_prob_plot': 'norm_prob_plot.png',
1804                        'CI_plot': 'simultaneous_CI_plot.png',
1805                        'percent': '\%'}
1806
1807        if use_summary:
1808            content_dict['summary'] = settings['summary']
1809
1810        content_dict.update(line.get_result_dict(design_lines, self.parent.get_n_sd(),
1811                                                 use_label=settings['use_line_name']))
1812
1813        for i, line in enumerate(manual_lines):
1814            line_key = 'manual_line'+str(i)
1815            line, name = line.get_line_info()
1816            if ' ' in name:
1817                name = name.replace(' ', '\ ')
1818            if '_' in name:
1819                name = name.replace('_', '\_')
1820            content_dict[line_key] = line % name
1821
1822        content_dict['linear_report'] = content_dict['linear_report'].replace('%', '\%')
1823
1824        if content_dict['linear_report'] == 'Could not test for linearity':
1825            message = '''Could not test for linearity. Stress range needs to be equal at each level,
1826    and the test needs at least 3 different levels.'''
1827            self.parent.on_error(message)
1828
1829        for i, (sigma, num) in enumerate(zip(sigma_values, n_values)):
1830            s_key = 's' + str(i)
1831            n_key = 'n' + str(i)
1832            content_dict[s_key] = sigma
1833            content_dict[n_key] = num
1834
1835        report.create_document(content_dict)
1836
1837    def report_to_txt(self, folder_path):
1838        sigma_values, n_values = self.parent.data_tab.get_data()
1839        line = self.parent.get_line()
1840        settings = self.parent.get_settings()
1841
1842        design_lines = []
1843        for line_name, active in self.parent.active_methods.iteritems():
1844            if active and line_name != 'reg_line':
1845                if 'iso' in line_name:
1846                    conf = self.parent.get_iso_conf()
1847                    prob = self.parent.get_iso_prob()
1848                    name = 'iso_'+conf+'_'+prob
```

```
                design_lines.append(name)
            else:
                design_lines.append(line_name)

        if self.parent.get_use_manual_lines():
            manual_lines = self.parent.get_manual_lines()
        else:
            manual_lines = []

        report_name = self.parent.get_settings()['project_name']+'_report'

        use_summary = settings['use_summary']
        report = ReportGeneratorTXT(folder_path, report_name)
        report.make_template(len(sigma_values), len(design_lines), len(manual_lines), use_summary)

        date = time.strftime("%d/%m/%Y")

        content_dict = {'title': settings['title'],
                        'author': settings['author'],
                        'date': date,
                        'intro': settings['intro'],
                        'percent': '%'}

        if use_summary:
            content_dict['summary'] = settings['summary']

        content_dict.update(line.get_result_dict(design_lines, self.parent.get_n_sd(),
                                                 use_label=settings['use_line_name'], txt=True))

        for i, line in enumerate(manual_lines):
            line_key = 'manual_line'+str(i)
            line, name = line.get_line_info_plain()
            if ' ' in name:
                name = name.replace(' ', '\ ')
            if '_' in name:
                name = name.replace('_', '\_')
            content_dict[line_key] = line % name

        if content_dict['linear_report'] == 'Could not test for linearity':
            message = '''Could not test for linearity. Stress range needs to be equal at each level,
and the test needs at least 3 different levels.'''
            self.parent.on_error(message)

        for i, (sigma, num) in enumerate(zip(sigma_values, n_values)):
            s_key = 's' + str(i)
            n_key = 'n' + str(i)
            content_dict[s_key] = sigma
            content_dict[n_key] = num

        report.create_document(content_dict)

    def clear(self):
        self.update_radio_btn()
        self.update_checkbox()
        self.axes.clear()
        self.canvas.draw()
        self.results_print.SetLabel('')


# noinspection PyBroadException
class CompareTab(wx.Panel):
    SETTINGS = {'project_name': 'Compare_S-N_lines', 'line1_name': 'Line 1', 'line2_name': 'Line 2',
                'y_lim': (1e2, 1e3), 'x_lim': (1e4, 1e7), 'report_format': '.pdf',
                'title': 'Comparing two datasets for S-N curves',
                'author': 'NMBU', 'intro': '', 'use_summary': False, 'summary': '', 'dpi': '300'}

    METHODS_CHECKED = {'design_2_sd_cb': False, 'design_iiw_cb': False,
                       'design_dnv_known_cb': False, 'design_dnv_unknown_cb': False, 'design_iso_cb': False,
                       'design_n_sd_cb': False}

    ACTIVE_METHODS = {'design_2_sd': False, 'design_iiw': False,
                      'design_dnv_known': False, 'design_dnv_unknown': False, 'design_iso': False,
                      'design_n_sd': False}

    ISO_OPTIONS = {'conf_90': True, 'conf_95': False, 'prob_10': False, 'prob_05': True, 'prob_01': False}

    N_SD = 3.0

    def __init__(self, parent):
        wx.Panel.__init__(self, parent=parent, id=wx.ID_ANY)
        self.parent = parent
        self.delimiter = self.parent.delimiter
        font = wx.Font(12, wx.DECORATIVE, wx.NORMAL, wx.BOLD)

        self.new_line_imported = False

        self.plot_styles = ['--', ':', '-.', 'v-', 'v--', 'v:', 'v-.']

```

```
1937        self.settings = self.SETTINGS.copy()
1938        self.methods_checked = self.METHODS_CHECKED.copy()
1939        self.active_methods = self.ACTIVE_METHODS.copy()
1940        self.iso_options = self.ISO_OPTIONS.copy()
1941
1942        self.n_sd_value = self.N_SD
1943
1944        self.figure = Figure((10, 10), dpi=72)
1945        self.canvas = FigureCanvas(self, -1, self.figure)
1946        self.axes = self.figure.add_subplot(111)
1947        self.toolbar = NavigationToolbar(self.canvas)
1948        self.figure.set_facecolor('#FFFFFF')
1949
1950        hsizer = wx.BoxSizer(wx.HORIZONTAL)
1951        vsizer = wx.BoxSizer(wx.VERTICAL)
1952        hsizerL12 = wx.BoxSizer(wx.HORIZONTAL)
1953        vsizerL1 = wx.BoxSizer(wx.VERTICAL)
1954        vsizerL2 = wx.BoxSizer(wx.VERTICAL)
1955        vsizer2 = wx.BoxSizer(wx.VERTICAL)
1956
1957        self.line1 = Line()
1958        self.line2 = Line()
1959
1960        header = wx.StaticText(self, -1, 'Compare:')
1961        header.SetFont(font)
1962        line_one_text = wx.StaticText(self, -1, 'Line 1:')
1963        line_two_text = wx.StaticText(self, -1, 'Line 2:')
1964        self.line1_info = wx.StaticText(self, -1, '')
1965        self.line2_info = wx.StaticText(self, -1, '')
1966        import_btn1 = wx.Button(self, wx.ID_FILE1, label='Import data', size=(100, 40))
1967        import_btn1.myname = 'line1'
1968        self.Bind(wx.EVT_BUTTON, self.on_import, import_btn1)
1969        import_btn2 = wx.Button(self, wx.ID_FILE2, label='Import data', size=(100, 40))
1970        import_btn2.myname = 'line2'
1971        self.Bind(wx.EVT_BUTTON, self.on_import, import_btn2)
1972
1973        settings_text = wx.StaticText(self, -1, 'Show:')
1974        self.show_points_cb = wx.CheckBox(self, label='Data points')
1975        self.design_2_sd_cb = wx.CheckBox(self, label='2 standard deviations (SD)')
1976        self.design_iiw_cb = wx.CheckBox(self, label='IIW')
1977        self.design_dnv_known_cb = wx.CheckBox(self, label='DNV, SD known')
1978        self.design_dnv_unknown_cb = wx.CheckBox(self, label='DNV, SD unknown')
1979        self.design_iso_cb = wx.CheckBox(self, label='ISO (simplified)')
1980        hsizer_confidence_txt = wx.BoxSizer(wx.HORIZONTAL)
1981        iso_text_conf = wx.StaticText(self, -1, 'Confidence level:')
1982        hsizer_confidence_txt.Add(iso_text_conf, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
1983        hsizer_confidence = wx.BoxSizer(wx.HORIZONTAL)
1984        self.conf_90 = wx.RadioButton(self, label='90%', style=wx.RB_GROUP)
1985        self.conf_95 = wx.RadioButton(self, label='95%')
1986        hsizer_confidence.Add(self.conf_90, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
1987        hsizer_confidence.Add(self.conf_95, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
1988        hsizer_probability_txt = wx.BoxSizer(wx.HORIZONTAL)
1989        iso_text_prob = wx.StaticText(self, -1, 'Probability:')
1990        hsizer_probability_txt.Add(iso_text_prob, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
1991        hsizer_probability = wx.BoxSizer(wx.HORIZONTAL)
1992        self.prob_10 = wx.RadioButton(self, label='10%', style=wx.RB_GROUP)
1993        self.prob_05 = wx.RadioButton(self, label='5%')
1994        self.prob_01 = wx.RadioButton(self, label='1%')
1995        hsizer_probability.Add(self.prob_10, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
1996        hsizer_probability.Add(self.prob_05, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
1997        hsizer_probability.Add(self.prob_01, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
1998        self.design_n_sd_cb = wx.CheckBox(self, label='n SD')
1999        self.n_sd = wx.TextCtrl(self, value=str(self.n_sd_value))
2000        self.n_sd.Disable()
2001
2002        settings_btn = wx.Button(self, wx.ID_ANY, label='Settings', size=(100, 40))
2003        self.Bind(wx.EVT_BUTTON, self.on_settings, settings_btn)
2004        compare_btn = wx.Button(self, wx.ID_ANY, label='Compare', size=(100, 60))
2005        self.Bind(wx.EVT_BUTTON, self.on_compare, compare_btn)
2006        report_btn = wx.Button(self, wx.ID_ANY, label='Create report', size=(100, 60))
2007        self.Bind(wx.EVT_BUTTON, self.on_report, report_btn)
2008
2009        vsizer.Add(header, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
2010        vsizerL1.Add(line_one_text, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
2011        vsizerL1.Add(self.line1_info, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
2012        vsizerL1.Add(import_btn1, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
2013        vsizerL2.Add(line_two_text, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
2014        vsizerL2.Add(self.line2_info, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
2015        vsizerL2.Add(import_btn2, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
2016        hsizerL12.Add(vsizerL1, 0)
2017        hsizerL12.Add(vsizerL2, 0)
2018        vsizer.Add(hsizerL12, 0)
2019
2020        vsizer.Add(settings_text, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
2021        vsizer.Add(self.show_points_cb, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
2022        vsizer.Add(self.design_2_sd_cb, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
2023        vsizer.Add(self.design_iiw_cb, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
2024        vsizer.Add(self.design_dnv_known_cb, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
```

```
2025        vsizer.Add(self.design_dnv_unknown_cb, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
2026        vsizer.Add(self.design_iso_cb, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
2027        vsizer.Add(hsizer_confidence_txt, 0, flag=wx.EXPAND | wx.LEFT, border=20)
2028        vsizer.Add(hsizer_confidence, 0, flag=wx.EXPAND | wx.LEFT, border=20)
2029        vsizer.Add(hsizer_probability_txt, 0, flag=wx.EXPAND | wx.LEFT, border=20)
2030        vsizer.Add(hsizer_probability, 0, flag=wx.EXPAND | wx.LEFT, border=20)
2031        vsizer.Add(self.design_n_sd_cb, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
2032        vsizer.Add(self.n_sd, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
2033
2034        vsizer.Add((-1, -1), -1, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2035        vsizer.Add(settings_btn, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2036        vsizer.Add(compare_btn, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2037        vsizer.Add(report_btn, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2038
2039        results_text = wx.StaticText(self, -1, 'Results:')
2040        results_text.SetFont(font)
2041        self.results_print = wx.StaticText(self, -1, '')
2042        vsizer2.Add(results_text, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
2043        vsizer2.Add(self.results_print, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
2044        vsizer2.Add(self.canvas, 1, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
2045        vsizer2.Add(self.toolbar, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)
2046
2047        hsizer.Add(vsizer, 0, wx.LEFT | wx.TOP | wx.BOTTOM | wx.GROW)
2048        hsizer.Add(vsizer2, -1, wx.LEFT | wx.TOP | wx.BOTTOM | wx.GROW)
2049        self.SetSizer(hsizer)
2050
2051        self.update_radio_btn()
2052        self.Bind(wx.EVT_CHECKBOX, self.on_check)
2053        self.Bind(wx.EVT_RADIOBUTTON, self.on_radio_btn)
2054
2055    def update_checkbox(self):
2056        for box, value in self.methods_checked.items():
2057            getattr(self, box).SetValue(value)
2058
2059    def on_check(self, event=None):
2060        for box, _ in self.methods_checked.items():
2061            value = getattr(self, box).GetValue()
2062            self.methods_checked[box] = value
2063            self.active_methods[box[:-3]] = value
2064        if self.design_n_sd_cb.GetValue():
2065            self.n_sd.Enable()
2066        else:
2067            self.n_sd.Disable()
2068
2069    def on_radio_btn(self, event=None):
2070        for btn, _ in self.iso_options.items():
2071            value = getattr(self, btn).GetValue()
2072            self.iso_options[btn] = value
2073
2074    def update_radio_btn(self):
2075        for btn, value in self.iso_options.items():
2076            getattr(self, btn).SetValue(value)
2077
2078    def get_settings(self):
2079        return self.settings
2080
2081    def set_settings(self, settings):
2082        self.settings = settings
2083
2084    def get_iso_conf(self):
2085        for key, value in self.iso_options.iteritems():
2086            if 'conf' in key and value:
2087                parts = key.split('_')
2088                return parts[1]
2089
2090    def get_iso_prob(self):
2091        for key, value in self.iso_options.iteritems():
2092            if 'prob' in key and value:
2093                parts = key.split('_')
2094                return parts[1]
2095
2096    def on_import(self, event=None):
2097        line = event.GetEventObject().myname
2098        wildcard = "Accepted file formats (*.csv; *.txt)|*.csv;*.txt"
2099        dialog = wx.FileDialog(self, 'Import data file', wildcard=wildcard, style=wx.OPEN)
2100        if dialog.ShowModal() == wx.ID_OK:
2101            path = dialog.GetPath()
2102            self.import_data(path, line)
2103            self.new_line_imported = True
2104            dialog.Destroy()
2105
2106    def import_data(self, path, line):
2107        file_name, file_format = os.path.splitext(path)
2108        sigma_values = []
2109        n_values = []
2110        with open(path, 'r') as infile:
2111            for data in infile.readlines():
2112                try:
```

```
                sigma, n = data.split(self.delimiter[file_format])
                sigma_values.append(eval(sigma))
                n_values.append(eval(n))
            except:
                pass  # header or text value
        getattr(self, line).update_values(n_values, sigma_values)
        getattr(self, line).calculate()

        line_info = line + '_info'
        getattr(self, line_info).SetLabel(self.make_line_info(getattr(self, line)))
        self.GetSizer().Layout()

    def on_compare(self, evt=None):
        if self.line1.sigma_values and self.line2.sigma_values:
            if self.new_line_imported:
                compare = CompareLines(self.line1, self.line2)
                results = compare.get_results()
                self.results_print.SetLabel(results)
                self.Layout()
                self.new_line_imported = False

                sigma_values = self.line1.sigma_values + self.line2.sigma_values
                sigma_min = int(np.min(sigma_values))
                n_dig = len(str(sigma_min))
                correction = 10.**(n_dig-1)
                y_min = m.floor(sigma_min/correction)*correction
                if y_min == sigma_min:
                    y_min -= correction

                sigma_max = int(np.max(sigma_values))
                n_dig = len(str(sigma_max))
                correction = 10.**(n_dig-1)
                y_max = m.ceil(sigma_max/correction)*correction
                if y_max == sigma_max:
                    y_max += correction
                self.settings['y_lim'] = (y_min, y_max)
            self.update_plot()

    def update_plot(self):
        self.axes.clear()
        legend = []

        n1, sigma1, label1, name = self.line1.get_line_points('reg_line')
        n2, sigma2, label2, name = self.line2.get_line_points('reg_line')
        self.axes.plot(n1, sigma1, 'b-')
        name1 = self.settings['line1_name']
        if ' ' in name1:
            name1 = name1.replace(' ', '\ ')
        if '_' in name1:
            name1 = name1.replace('_', '\_')
        name2 = self.settings['line2_name']
        if ' ' in name2:
            name2 = name2.replace(' ', '\ ')
        if '_' in name2:
            name2 = name2.replace('_', '\_')
        label1 = label1 % name1
        label2 = label2 % name2
        legend.append(label1)
        self.axes.plot(n2, sigma2, 'r-')
        legend.append(label2)

        show_points = self.show_points_cb.GetValue()
        if show_points:
            s1_values, n1_values = self.line1.get_data()
            self.axes.plot(n1_values, s1_values, 'bo')
            legend.append(self.settings['line1_name']+' Datapoints')
            s2_values, n2_values = self.line2.get_data()
            self.axes.plot(n2_values, s2_values, 'ro')
            legend.append(self.settings['line2_name']+' Datapoints')
        n_design = 0
        for method, use in self.active_methods.items():
            if use:
                if 'n_sd' in method:
                    n_sd = float(self.n_sd.GetValue())
                    n1, sigma1, label1, name1 = self.line1.get_line_points(method, n_sd)
                    n2, sigma2, label2, name2 = self.line2.get_line_points(method, n_sd)
                elif 'iso' in method:
                    conf = self.get_iso_conf()
                    prob = self.get_iso_prob()
                    name = 'iso_'+conf+'_'+prob
                    n1, sigma1, label1, name1 = self.line1.get_line_points(name)
                    n2, sigma2, label2, name2 = self.line2.get_line_points(name)
                else:
                    n1, sigma1, label1, name1 = self.line1.get_line_points(method)
                    n2, sigma2, label2, name2 = self.line2.get_line_points(method)

                self.axes.plot(n1, sigma1, 'b'+self.plot_styles[n_design])
                legend.append(label1 % name1)
```

```
            self.axes.plot(n2, sigma2, 'r'+self.plot_styles[n_design])
            legend.append(label2 % name2)
            n_design += 1

        self.axes.legend(legend)

        self.axes.set_yscale('log')
        self.axes.set_xscale('log')
        self.axes.set_ylim(self.settings['y_lim'])
        self.axes.set_xlim(self.settings['x_lim'])
        self.axes.yaxis.set_minor_formatter(FormatStrFormatter("%d"))

        self.axes.grid(True, which='major', linestyle='--')
        self.axes.grid(True, which='minor', linestyle='--')

        font = {'weight': 'normal',
                'size': 18}

        self.axes.set_xlabel(r'Number of Cycles [N]', fontdict=font)
        self.axes.set_ylabel(r'Stress Range, $\Delta \sigma $ [MPa]', fontdict=font)

        self.canvas.draw()

    def save_plot(self, path):
        name = path + '\\' + 'compare_lines'
        self.figure.savefig(name, dpi=float(self.settings['dpi']))
        self.update_plot()  # Reset canvas

    def save_data(self, line, path):
        if '.txt' in path:
            file_format = '.txt'
        elif '.csv' in path:
            file_format = '.csv'
        else:
            file_format = '.txt'
            path += file_format
        sigma_values, n_values = line.get_data()
        header = 'Sigma' + self.delimiter[file_format] + 'N'

        with open(path, 'w') as infile:
            infile.write(header)
            for sigma, N in zip(sigma_values, n_values):
                infile.write('\n' + str(sigma) + self.delimiter[file_format] + str(N))

    def update_line_value(self):
        sigma_values, n_values = self.parent.get_line().get_data()
        self.line1.update_values(n_values, sigma_values)
        self.line1_info.SetLabel(self.make_line_info(self.line1))
        self.GetSizer().Layout()

    @staticmethod
    def make_line_info(line):
        info = ''
        info += '    m: {:.2f}\n'.format(line.get_m())
        info += '    c: {:.2e}\n'.format(line.get_c())
        info += '    sd: {:.4f}\n'.format(line.get_sd())
        info += '    N: {:}\n'.format(line.get_num_values())
        return info

    def on_settings(self, event=None):
        CompareSettingWindow(self)

    def on_report(self, event=None):
        if self.line1.sigma_values and self.line2.sigma_values:
            dialog = wx.DirDialog(self, ' Choose a file location', style=wx.SAVE)
            if dialog.ShowModal() == wx.ID_OK:
                path = dialog.GetPath()
                self.report(path)
                dialog.Destroy()
        else:
            PopupWindow(self, 'Reporting terminated', 'No data detected', size=(300, 200))

    def report(self, path):
        settings = self.settings
        self.on_compare()
        foldername = settings['project_name']+'_report'
        folder_num = 1

        for folder in [folder for folder in os.listdir(path) if os.path.isdir(os.path.join(path, folder))]:
            if foldername in folder and '_' in folder:
                elements = folder.split('_')
                try:
                    number = int(elements[-1])
                    if folder_num <= number:
                        folder_num = number + 1
                except:
                    pass
        folder_path = path + '\\' + foldername + '_' + str(folder_num)
```

```
        os.makedirs(folder_path)

        self.save_plot(folder_path)
        self.save_data(self.line1, folder_path + '\\' + settings['line1_name'] + '_data.csv')
        self.save_data(self.line2, folder_path + '\\' + settings['line2_name'] + '_data.csv')

        if settings['report_format'] == '.pdf':
            self.report_to_pdf(folder_path)
        else:
            self.report_to_txt(folder_path)
        PopupWindow(self, '', 'Report done', size=(300, 200))

    def report_to_pdf(self, folder_path):
        line1 = self.line1
        line2 = self.line2
        settings = self.settings

        design_lines = []
        for line_name, active in self.active_methods.iteritems():
            if active and line_name != 'reg_line':
                if 'iso' in line_name:
                    conf = self.get_iso_conf()
                    prob = self.get_iso_prob()
                    name = 'iso_'+conf+'_'+prob
                    design_lines.append(name)
                else:
                    design_lines.append(line_name)

        report_name = self.settings['project_name']+'_report'

        use_summary = settings['use_summary']
        report = CompareReportGenerator(folder_path, report_name)
        report.make_template(len(line1.sigma_values), len(line2.sigma_values), len(design_lines), use_summary)

        name1 = self.settings['line1_name']
        if ' ' in name1:
            name1 = name1.replace(' ', '\ ')
        if '_' in name1:
            name1 = name1.replace('_', '\_')
        name2 = self.settings['line2_name']
        if ' ' in name2:
            name2 = name2.replace(' ', '\ ')
        if '_' in name2:
            name2 = name2.replace('_', '\_')

        content_dict = {'title': settings['title'],
                        'author': settings['author'],
                        'intro': settings['intro'],
                        'compare_plot': 'compare_lines.png',
                        'percent': '\%',
                        'name1': name1,
                        'name2': name2}

        if use_summary:
            content_dict['summary'] = settings['summary']

        temp_dict = {}
        n_sd = float(self.n_sd.GetValue())
        print design_lines
        for key, value in line1.get_lines_with_name(design_lines, n_sd, use_label=False).iteritems():
            if key == 'reg_line':
                key = 'line'
                temp_dict[key+'_1'] = value
            else:
                temp_dict[key+'_1'] = value[0]
                if '%.1f' in value[1]:
                    temp_dict[key] = value[1] % n_sd
                else:
                    temp_dict[key] = value[1]

        content_dict.update(temp_dict)
        temp_dict = {}
        for key, value in line2.get_lines_with_name(design_lines, n_sd, use_label=False).iteritems():
            if key == 'reg_line':
                key = 'line'
                temp_dict[key+'_2'] = value
            else:
                temp_dict[key+'_2'] = value[0]
        content_dict.update(temp_dict)

        for i, (sigma, num) in enumerate(zip(line1.sigma_values, line1.n_values)):
            s_key = 's1' + str(i)
            n_key = 'n1' + str(i)
            content_dict[s_key] = sigma
            content_dict[n_key] = num
        for i, (sigma, num) in enumerate(zip(line2.sigma_values, line2.n_values)):
            s_key = 's2' + str(i)
            n_key = 'n2' + str(i)
```

```
2377            content_dict[s_key] = sigma
2378            content_dict[n_key] = num
2379
2380        compare = CompareLines(self.line1, self.line2)
2381        results = compare.get_results_double_lines()
2382        while ' %' in results:
2383            results = results.replace(' %', ' \%')
2384        content_dict['results'] = results
2385
2386        report.create_document(content_dict)
2387
2388    def report_to_txt(self, folder_path):
2389        line1 = self.line1
2390        line2 = self.line2
2391        settings = self.settings
2392
2393        design_lines = []
2394        for line_name, active in self.active_methods.iteritems():
2395            if active and line_name != 'reg_line':
2396                if 'iso' in line_name:
2397                    conf = self.get_iso_conf()
2398                    prob = self.get_iso_prob()
2399                    name = 'iso_'+conf+'_'+prob
2400                    design_lines.append(name)
2401                else:
2402                    design_lines.append(line_name)
2403
2404        report_name = self.settings['project_name']+'_report'
2405
2406        use_summary = settings['use_summary']
2407        report = CompareReportGeneratorTXT(folder_path, report_name)
2408        report.make_template(len(line1.sigma_values), len(line2.sigma_values), len(design_lines), use_summary)
2409
2410        date = time.strftime("%d/%m/%Y")
2411
2412        content_dict = {'title': settings['title'],
2413                        'author': settings['author'],
2414                        'date': date,
2415                        'intro': settings['intro'],
2416                        'percent': '%',
2417                        'blank': ' ',
2418                        'name1': settings['line1_name'],
2419                        'name2': settings['line2_name']}
2420
2421        if use_summary:
2422            content_dict['summary'] = settings['summary']
2423
2424        n_sd = float(self.n_sd.GetValue())
2425        temp_dict = {}
2426        for key, value in line1.get_lines_with_name(design_lines, n_sd, use_label=False, txt=True).iteritems():
2427            if key == 'reg_line':
2428                key = 'line'
2429                temp_dict[key+'_1'] = value
2430            else:
2431                temp_dict[key+'_1'] = value[0]
2432                if '%.1f' in value[1]:
2433                    temp_dict[key] = value[1] % n_sd
2434                else:
2435                    temp_dict[key] = value[1]
2436
2437        content_dict.update(temp_dict)
2438        temp_dict = {}
2439        for key, value in line2.get_lines_with_name(design_lines, n_sd, use_label=False, txt=True).iteritems():
2440            if key == 'reg_line':
2441                key = 'line'
2442                temp_dict[key+'_2'] = value
2443            else:
2444                temp_dict[key+'_2'] = value[0]
2445        content_dict.update(temp_dict)
2446
2447        for i, (sigma, num) in enumerate(zip(line1.sigma_values, line1.n_values)):
2448            s_key = 's1' + str(i)
2449            n_key = 'n1' + str(i)
2450            content_dict[s_key] = sigma
2451            content_dict[n_key] = num
2452        for i, (sigma, num) in enumerate(zip(line2.sigma_values, line2.n_values)):
2453            s_key = 's2' + str(i)
2454            n_key = 'n2' + str(i)
2455            content_dict[s_key] = sigma
2456            content_dict[n_key] = num
2457
2458        compare = CompareLines(self.line1, self.line2)
2459        results = compare.get_results_double_lines()
2460        content_dict['results'] = results
2461
2462        report.create_document(content_dict)
2463
2464    def clear(self):
```

```
2465        self.settings = self.SETTINGS.copy()
2466        self.methods_checked = self.METHODS_CHECKED.copy()
2467        self.active_methods = self.ACTIVE_METHODS.copy()
2468        self.iso_options = self.ISO_OPTIONS.copy()
2469
2470        self.n_sd.SetValue(str(self.N_SD))
2471        self.update_radio_btn()
2472        self.update_checkbox()
2473        self.show_points_cb.SetValue(False)
2474
2475        self.axes.clear()
2476        self.line1_info.SetLabel('')
2477        self.line2_info.SetLabel('')
2478        self.results_print.SetLabel('')
2479        self.Layout()
2480
2481
2482    class PopupWindow(wx.PopupWindow):
2483        def __init__(self, parent, message, header=None, size=(500, 300)):
2484            wx.PopupWindow.__init__(self, parent, wx.SIMPLE_BORDER)
2485            self.SetSize(size)
2486
2487            default_font = wx.Font(12, wx.DEFAULT, wx.NORMAL, wx.NORMAL)
2488            self.SetFont(default_font)
2489
2490            width = size[0]
2491            height = size[1]
2492
2493            if header:
2494                font = wx.Font(12, wx.DECORATIVE, wx.NORMAL, wx.BOLD)
2495                header_text = header
2496                x_header = wx.TextCtrl(self, -1, header_text, pos=(20, 20),
2497                                       style=wx.TE_READONLY | wx.BORDER_NONE)
2498                x_header.SetFont(font)
2499
2500                text = wx.TextCtrl(self, -1, message, size=(width - 40, height - 110), pos=(20, 50),
2501                                   style=wx.TE_READONLY | wx.TE_MULTILINE | wx.BORDER_NONE)
2502            else:
2503                text = wx.TextCtrl(self, -1, message, size=(width - 40, height - 80), pos=(20, 20),
2504                                   style=wx.TE_READONLY | wx.TE_MULTILINE | wx.BORDER_NONE)
2505            text.SetBackgroundColour('#f0f0f0')
2506            close_btn = wx.Button(self, wx.ID_CLOSE, label='Close', pos=(width - 120, height - 60), size=(100, 40))
2507            self.Bind(wx.EVT_BUTTON, self.on_close, close_btn)
2508
2509            self.CentreOnParent(dir=wx.BOTH)
2510            self.Show()
2511
2512        def on_close(self, _):
2513            self.Destroy()
2514
2515
2516    class SettingWindow(wx.Frame):
2517        def __init__(self, parent):
2518            super(SettingWindow, self).__init__(parent, title='Settings', size=(300, 800))
2519            self.parent = parent
2520            self.settings = self.parent.get_settings()
2521
2522            default_font = wx.Font(12, wx.DEFAULT, wx.NORMAL, wx.NORMAL)
2523            self.SetFont(default_font)
2524
2525            icon = wx.Icon('graphics/T_logo.ico', wx.BITMAP_TYPE_ICO)
2526            self.SetIcon(icon)
2527
2528            panel = wx.Panel(self, -1)
2529            vsizer = wx.BoxSizer(wx.VERTICAL)
2530            vsizer_outer = wx.BoxSizer(wx.VERTICAL)
2531
2532            done_btn = wx.Button(panel, wx.ID_CLOSE, label='Done', size=(100, 40))
2533
2534            font = wx.Font(12, wx.DECORATIVE, wx.NORMAL, wx.BOLD)
2535
2536            project_header_text = 'Project'
2537            hsizer_project1 = wx.BoxSizer(wx.HORIZONTAL)
2538            priject_header = wx.StaticText(panel, -1, project_header_text)
2539            priject_header.SetFont(font)
2540            hsizer_project1.Add(priject_header)
2541            vsizer.Add(hsizer_project1, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2542
2543            hsizer_project2 = wx.BoxSizer(wx.HORIZONTAL)
2544            project_text = wx.StaticText(panel, -1, 'Project name:')
2545            hsizer_project2.Add(project_text, -1)
2546            vsizer.Add(hsizer_project2, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2547
2548            hsizer_project3 = wx.BoxSizer(wx.HORIZONTAL)
2549            project_name = self.settings['project_name'].decode('utf-8')
2550            self.projectname = wx.TextCtrl(panel, value=project_name)
2551            hsizer_project3.Add(self.projectname, -1)
2552            vsizer.Add(hsizer_project3, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
```

```
2553
2554        hsizer_project2 = wx.BoxSizer(wx.HORIZONTAL)
2555        project_text = wx.StaticText(panel, -1, 'Line name:')
2556        hsizer_project2.Add(project_text, -1)
2557        vsizer.Add(hsizer_project2, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2558
2559        hsizer_project3 = wx.BoxSizer(wx.HORIZONTAL)
2560        line_name = self.settings['line_name'].decode('utf-8')
2561        self.line_name = wx.TextCtrl(panel, value=line_name)
2562        hsizer_project3.Add(self.line_name, -1)
2563        vsizer.Add(hsizer_project3, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2564        hsizer_project4 = wx.BoxSizer(wx.HORIZONTAL)
2565        self.use_line_name = wx.CheckBox(panel, label='Use line name on design lines')
2566        self.use_line_name.SetValue(self.settings['use_line_name'])
2567        hsizer_project4.Add(self.use_line_name)
2568        vsizer.Add(hsizer_project4, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2569        vsizer.Add((-1, 20))
2570
2571        hsizer_results = wx.BoxSizer(wx.HORIZONTAL)
2572        results_header = wx.StaticText(panel, -1, 'Results')
2573        results_header.SetFont(font)
2574        hsizer_results.Add(results_header)
2575        vsizer.Add(hsizer_results, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2576
2577        hsizer_results0 = wx.BoxSizer(wx.HORIZONTAL)
2578        format_text = wx.StaticText(panel, -1, 'Report format:')
2579        hsizer_results0.Add(format_text)
2580        vsizer.Add(hsizer_results0, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2581
2582        hsizer_results1 = wx.BoxSizer(wx.HORIZONTAL)
2583        self.report_format_pdf = wx.RadioButton(panel, label='.pdf', style=wx.RB_GROUP)
2584        self.report_format_txt = wx.RadioButton(panel, label='.txt')
2585        if self.settings['report_format'] == '.pdf':
2586            self.report_format_pdf.SetValue(True)
2587        else:
2588            self.report_format_txt.SetValue(True)
2589        hsizer_results1.Add(self.report_format_pdf, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
2590        hsizer_results1.Add(self.report_format_txt, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
2591        vsizer.Add(hsizer_results1)
2592        vsizer.Add((-1, 10))
2593
2594        hsizer_results2 = wx.BoxSizer(wx.HORIZONTAL)
2595        title_text = wx.StaticText(panel, -1, 'Title:')
2596        hsizer_results2.Add(title_text)
2597        vsizer.Add(hsizer_results2, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2598
2599        hsizer_results3 = wx.BoxSizer(wx.HORIZONTAL)
2600        title = self.settings['title'].decode('utf-8')
2601        self.title = wx.TextCtrl(panel, value=title)
2602        hsizer_results3.Add(self.title, -1)
2603        vsizer.Add(hsizer_results3, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2604        vsizer.Add((-1, 10))
2605
2606        hsizer_results4 = wx.BoxSizer(wx.HORIZONTAL)
2607        author_text = wx.StaticText(panel, -1, 'Author:')
2608        hsizer_results4.Add(author_text, -1)
2609        vsizer.Add(hsizer_results4, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2610
2611        hsizer_results5 = wx.BoxSizer(wx.HORIZONTAL)
2612        author = self.settings['author'].decode('utf-8')
2613        self.author = wx.TextCtrl(panel, value=author)
2614        hsizer_results5.Add(self.author, -1)
2615        vsizer.Add(hsizer_results5, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2616        vsizer.Add((-1, 10))
2617
2618        hsizer_results6 = wx.BoxSizer(wx.HORIZONTAL)
2619        intro_text = wx.StaticText(panel, -1, 'Introduction:')
2620        hsizer_results6.Add(intro_text)
2621        vsizer.Add(hsizer_results6, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2622
2623        hsizer_results7 = wx.BoxSizer(wx.HORIZONTAL)
2624        self.intro = wx.TextCtrl(panel, size=(280, 100), style=wx.TE_MULTILINE)
2625        intro = self.settings['intro'].decode('utf-8')
2626        self.intro.SetValue(intro)
2627        hsizer_results7.Add(self.intro, -1)
2628        vsizer.Add(hsizer_results7, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2629        vsizer.Add((-1, 10))
2630
2631        hsizer_results8 = wx.BoxSizer(wx.HORIZONTAL)
2632        summary_text = wx.StaticText(panel, -1, 'Summary:')
2633        hsizer_results8.Add(summary_text)
2634        vsizer.Add(hsizer_results8, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2635
2636        hsizer_results9 = wx.BoxSizer(wx.HORIZONTAL)
2637        self.use_summary = wx.CheckBox(panel, label='Add summary')
2638        self.use_summary.SetValue(self.settings['use_summary'])
2639        hsizer_results9.Add(self.use_summary)
2640        vsizer.Add(hsizer_results9, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
```

```
2641
2642        hsizer_results10 = wx.BoxSizer(wx.HORIZONTAL)
2643        self.summary = wx.TextCtrl(panel, size=(280, 100), style=wx.TE_MULTILINE)
2644        summary = self.settings['summary'].decode('utf-8')
2645        self.summary.SetValue(summary)
2646        if not self.settings['use_summary']:
2647            self.summary.Disable()
2648        hsizer_results10.Add(self.summary, -1)
2649        vsizer.Add(hsizer_results10, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2650        vsizer.Add((-1, 10))
2651
2652        hsizer_results11 = wx.BoxSizer(wx.HORIZONTAL)
2653        format_text = wx.StaticText(panel, -1, 'Figure resolution (dpi):')
2654        hsizer_results11.Add(format_text)
2655        vsizer.Add(hsizer_results11, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2656
2657        hsizer_results12 = wx.BoxSizer(wx.HORIZONTAL)
2658        self.dpi_72 = wx.RadioButton(panel, label='72', style=wx.RB_GROUP)
2659        self.dpi_150 = wx.RadioButton(panel, label='150')
2660        self.dpi_300 = wx.RadioButton(panel, label='300')
2661        self.dpi_600 = wx.RadioButton(panel, label='600')
2662        active_dpi_name = 'dpi_' + self.settings['dpi']
2663        self.dpi_values = {'72': self.dpi_72, '150': self.dpi_150, '300': self.dpi_300, '600': self.dpi_600}
2664        getattr(self, active_dpi_name).SetValue(True)
2665
2666        hsizer_results12.Add(self.dpi_72, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
2667        hsizer_results12.Add(self.dpi_150, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
2668        hsizer_results12.Add(self.dpi_300, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
2669        hsizer_results12.Add(self.dpi_600, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
2670        vsizer.Add(hsizer_results12)
2671        vsizer.Add((-1, 30))
2672
2673        self.Bind(wx.EVT_BUTTON, self.on_done, done_btn)
2674        vsizer.Add((1, 1), -1, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2675        vsizer.Add(done_btn, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2676
2677        self.Bind(wx.EVT_CHECKBOX, self.on_check)
2678
2679        vsizer_outer.Add(vsizer, -1, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP | wx.BOTTOM, border=10)
2680        panel.SetSizer(vsizer_outer)
2681
2682        self.Centre()
2683        self.Show()
2684
2685    def on_done(self, _):
2686        if self.report_format_pdf.GetValue():
2687            report_format = '.pdf'
2688        else:
2689            report_format = '.txt'
2690        dpi_value = [key for key, value in self.dpi_values.items() if value.GetValue()][0]
2691        projectname = self.projectname.GetValue()
2692        if ' ' in projectname:
2693            projectname = projectname.replace(' ', '_')
2694        self.parent.set_settings({'project_name': projectname, 'line_name': self.line_name.GetValue(),
2695                                  'use_line_name': self.use_line_name.GetValue(),
2696                                  'report_format': report_format,
2697                                  'title': self.title.GetValue(), 'author': self.author.GetValue(),
2698                                  'intro': self.intro.GetValue(), 'use_summary': self.use_summary.GetValue(),
2699                                  'summary': self.summary.GetValue(), 'dpi': dpi_value})
2700        self.parent.set_line_label(self.line_name.GetValue())
2701        self.parent.set_line_use_label(self.use_line_name.GetValue())
2702        self.parent.on_run()
2703        self.Close()
2704
2705    def on_check(self, _):
2706        if self.use_summary.GetValue():
2707            self.summary.Enable()
2708        else:
2709            self.summary.Disable()
2710
2711
2712 class CompareSettingWindow(wx.Frame):
2713    def __init__(self, parent):
2714        super(CompareSettingWindow, self).__init__(parent, title='Settings', size=(400, 850))
2715        self.parent = parent
2716        self.settings = self.parent.get_settings()
2717
2718        default_font = wx.Font(12, wx.DEFAULT, wx.NORMAL, wx.NORMAL)
2719        self.SetFont(default_font)
2720
2721        icon = wx.Icon('graphics/T_logo.ico', wx.BITMAP_TYPE_ICO)
2722        self.SetIcon(icon)
2723
2724        panel = wx.Panel(self, -1)
2725        vsizer = wx.BoxSizer(wx.VERTICAL)
2726        vsizer_outer = wx.BoxSizer(wx.VERTICAL)
2727
2728        done_btn = wx.Button(panel, wx.ID_CLOSE, label='Done', size=(100, 40))
```

```
2729
2730        font = wx.Font(12, wx.DECORATIVE, wx.NORMAL, wx.BOLD)
2731
2732        project_header_text = 'Project'
2733        hsizer_project1 = wx.BoxSizer(wx.HORIZONTAL)
2734        project_header = wx.StaticText(panel, -1, project_header_text)
2735        project_header.SetFont(font)
2736        hsizer_project1.Add(project_header)
2737        vsizer.Add(hsizer_project1, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2738
2739        hsizer_project2 = wx.BoxSizer(wx.HORIZONTAL)
2740        project_text = wx.StaticText(panel, -1, 'Project name:')
2741        hsizer_project2.Add(project_text, -1)
2742        vsizer.Add(hsizer_project2, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2743
2744        hsizer_project3 = wx.BoxSizer(wx.HORIZONTAL)
2745        project_name = self.settings['project_name'].decode('utf-8')
2746        self.projectname = wx.TextCtrl(panel, value=project_name)
2747        hsizer_project3.Add(self.projectname, -1)
2748        vsizer.Add(hsizer_project3, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2749
2750        vsizer.Add((-1, 5))
2751        hsizer_l1 = wx.BoxSizer(wx.HORIZONTAL)
2752        label = wx.StaticText(panel, 0, 'Line 1 name:')
2753        line1_name = self.settings['line1_name'].decode('utf-8')
2754        self.line_name_1 = wx.TextCtrl(panel, value=line1_name)
2755        hsizer_l1.Add(label, 1)
2756        hsizer_l1.Add(self.line_name_1, 2)
2757
2758        hsizer_l2 = wx.BoxSizer(wx.HORIZONTAL)
2759        label = wx.StaticText(panel, 0, 'Line 2 name:')
2760        line2_name = self.settings['line2_name'].decode('utf-8')
2761        self.line_name_2 = wx.TextCtrl(panel, value=line2_name)
2762        hsizer_l2.Add(label, 1)
2763        hsizer_l2.Add(self.line_name_2, 2)
2764
2765        vsizer.Add(hsizer_l1, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2766        vsizer.Add((-1, 2))
2767        vsizer.Add(hsizer_l2, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2768        vsizer.Add((-1, 10))
2769
2770        y_header_text = 'Y-limits'
2771        hsizer_y1 = wx.BoxSizer(wx.HORIZONTAL)
2772        y_header = wx.StaticText(panel, -1, y_header_text)
2773        y_header.SetFont(font)
2774        x_header_text = 'X-limits'
2775        x_header = wx.StaticText(panel, -1, x_header_text)
2776        x_header.SetFont(font)
2777        hsizer_y1.Add(y_header, 1)
2778        hsizer_y1.Add(x_header, 1)
2779        vsizer.Add(hsizer_y1, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2780
2781        hsizer_y2 = wx.BoxSizer(wx.HORIZONTAL)
2782        ymin = wx.StaticText(panel, 0, 'y_min:')
2783        self.y_min = wx.TextCtrl(panel, value=str(self.settings['y_lim'][0]))
2784        xmin = wx.StaticText(panel, 0, 'x_min:')
2785        x_min_exp = '%.2e' % (self.settings['x_lim'][0])
2786        self.x_min = wx.TextCtrl(panel, value=str(x_min_exp))
2787        hsizer_y2.Add(ymin, 1)
2788        hsizer_y2.Add(self.y_min, 1)
2789        hsizer_y2.Add((5, 5))
2790        hsizer_y2.Add(xmin, 1)
2791        hsizer_y2.Add(self.x_min, 1)
2792
2793        hsizer_y3 = wx.BoxSizer(wx.HORIZONTAL)
2794        ymax = wx.StaticText(panel, 0, 'y_max:')
2795        self.y_max = wx.TextCtrl(panel, value=str(self.settings['y_lim'][1]))
2796        xmax = wx.StaticText(panel, 0, 'x_max:')
2797        x_max_exp = '%.2e' % (self.settings['x_lim'][1])
2798        self.x_max = wx.TextCtrl(panel, value=str(x_max_exp))
2799        hsizer_y3.Add(ymax, 1)
2800        hsizer_y3.Add(self.y_max, 1)
2801        hsizer_y3.Add((5, 5))
2802        hsizer_y3.Add(xmax, 1)
2803        hsizer_y3.Add(self.x_max, 1)
2804
2805        vsizer.Add(hsizer_y2, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2806        vsizer.Add((-1, 2))
2807        vsizer.Add(hsizer_y3, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2808        vsizer.Add((-1, 10))
2809
2810        hsizer_results = wx.BoxSizer(wx.HORIZONTAL)
2811        results_header = wx.StaticText(panel, -1, 'Results')
2812        results_header.SetFont(font)
2813        hsizer_results.Add(results_header)
2814        vsizer.Add(hsizer_results, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2815
2816        hsizer_results0 = wx.BoxSizer(wx.HORIZONTAL)
```

```
2817        format_text = wx.StaticText(panel, -1, 'Report format:')
2818        hsizer_results0.Add(format_text)
2819        vsizer.Add(hsizer_results0, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2820
2821        hsizer_results1 = wx.BoxSizer(wx.HORIZONTAL)
2822        self.report_format_pdf = wx.RadioButton(panel, label='.pdf', style=wx.RB_GROUP)
2823        self.report_format_txt = wx.RadioButton(panel, label='.txt')
2824        if self.settings['report_format'] == '.pdf':
2825            self.report_format_pdf.SetValue(True)
2826        else:
2827            self.report_format_txt.SetValue(True)
2828        hsizer_results1.Add(self.report_format_pdf, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
2829        hsizer_results1.Add(self.report_format_txt, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
2830        vsizer.Add(hsizer_results1)
2831        vsizer.Add((-1, 10))
2832
2833        hsizer_results2 = wx.BoxSizer(wx.HORIZONTAL)
2834        title_text = wx.StaticText(panel, -1, 'Title:')
2835        hsizer_results2.Add(title_text)
2836        vsizer.Add(hsizer_results2, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2837
2838        hsizer_results3 = wx.BoxSizer(wx.HORIZONTAL)
2839        title = self.settings['title'].decode('utf-8')
2840        self.title = wx.TextCtrl(panel, value=title)
2841        hsizer_results3.Add(self.title, -1)
2842        vsizer.Add(hsizer_results3, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2843        vsizer.Add((-1, 10))
2844
2845        hsizer_results4 = wx.BoxSizer(wx.HORIZONTAL)
2846        author_text = wx.StaticText(panel, -1, 'Author:')
2847        hsizer_results4.Add(author_text, -1)
2848        vsizer.Add(hsizer_results4, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2849
2850        hsizer_results5 = wx.BoxSizer(wx.HORIZONTAL)
2851        author = self.settings['author'].decode('utf-8')
2852        self.author = wx.TextCtrl(panel, value=author)
2853        hsizer_results5.Add(self.author, -1)
2854        vsizer.Add(hsizer_results5, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2855        vsizer.Add((-1, 10))
2856
2857        hsizer_results6 = wx.BoxSizer(wx.HORIZONTAL)
2858        intro_text = wx.StaticText(panel, -1, 'Introduction:')
2859        hsizer_results6.Add(intro_text)
2860        vsizer.Add(hsizer_results6, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2861
2862        hsizer_results7 = wx.BoxSizer(wx.HORIZONTAL)
2863        self.intro = wx.TextCtrl(panel, size=(280, 100), style=wx.TE_MULTILINE)
2864        intro = self.settings['intro'].decode('utf-8')
2865        self.intro.SetValue(intro)
2866        hsizer_results7.Add(self.intro, -1)
2867        vsizer.Add(hsizer_results7, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2868        vsizer.Add((-1, 10))
2869
2870        hsizer_results8 = wx.BoxSizer(wx.HORIZONTAL)
2871        summary_text = wx.StaticText(panel, -1, 'Summary:')
2872        hsizer_results8.Add(summary_text)
2873        vsizer.Add(hsizer_results8, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2874
2875        hsizer_results9 = wx.BoxSizer(wx.HORIZONTAL)
2876        self.use_summary = wx.CheckBox(panel, label='Add summary')
2877        self.use_summary.SetValue(self.settings['use_summary'])
2878        hsizer_results9.Add(self.use_summary)
2879        vsizer.Add(hsizer_results9, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2880
2881        hsizer_results10 = wx.BoxSizer(wx.HORIZONTAL)
2882        self.summary = wx.TextCtrl(panel, size=(280, 100), style=wx.TE_MULTILINE)
2883        summary = self.settings['summary'].decode('utf-8')
2884        self.summary.SetValue(summary)
2885        if not self.settings['use_summary']:
2886            self.summary.Disable()
2887        hsizer_results10.Add(self.summary, -1)
2888        vsizer.Add(hsizer_results10, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2889        vsizer.Add((-1, 10))
2890
2891        hsizer_results11 = wx.BoxSizer(wx.HORIZONTAL)
2892        format_text = wx.StaticText(panel, -1, 'Figure resolution (dpi):')
2893        hsizer_results11.Add(format_text)
2894        vsizer.Add(hsizer_results11, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2895
2896        hsizer_results12 = wx.BoxSizer(wx.HORIZONTAL)
2897        self.dpi_72 = wx.RadioButton(panel, label='72', style=wx.RB_GROUP)
2898        self.dpi_150 = wx.RadioButton(panel, label='150')
2899        self.dpi_300 = wx.RadioButton(panel, label='300')
2900        self.dpi_600 = wx.RadioButton(panel, label='600')
2901        active_dpi_name = 'dpi_' + self.settings['dpi']
2902        self.dpi_values = {'72': self.dpi_72, '150': self.dpi_150, '300': self.dpi_300, '600': self.dpi_600}
2903        getattr(self, active_dpi_name).SetValue(True)
2904
```

```
2905        hsizer_results12.Add(self.dpi_72, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
2906        hsizer_results12.Add(self.dpi_150, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
2907        hsizer_results12.Add(self.dpi_300, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
2908        hsizer_results12.Add(self.dpi_600, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=5)
2909        vsizer.Add(hsizer_results12)
2910        vsizer.Add((-1, 30))
2911
2912        self.Bind(wx.EVT_BUTTON, self.on_done, done_btn)
2913        vsizer.Add((1, 1), -1, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2914        vsizer.Add(done_btn, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2915
2916        self.Bind(wx.EVT_CHECKBOX, self.on_check)
2917
2918        vsizer_outer.Add(vsizer, -1, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP | wx.BOTTOM, border=10)
2919        panel.SetSizer(vsizer_outer)
2920
2921        self.Centre()
2922        self.Show()
2923
2924    def on_done(self, _):
2925        if self.report_format_pdf.GetValue():
2926            report_format = '.pdf'
2927        else:
2928            report_format = '.txt'
2929        dpi_value = [key for key, value in self.dpi_values.items() if value.GetValue()][0]
2930        projectname = self.projectname.GetValue()
2931        if ' ' in projectname:
2932            projectname = projectname.replace(' ', '_')
2933        if self.x_min.GetValue() == '0':
2934            self.x_min.SetValue('1')
2935        if self.y_min.GetValue() == '0':
2936            self.y_min.SetValue('1')
2937        y_lim = (eval(self.y_min.GetValue()), eval(self.y_max.GetValue()))
2938        x_lim = (eval(self.x_min.GetValue()), eval(self.x_max.GetValue()))
2939        self.parent.set_settings({'project_name': projectname, 'line1_name': self.line_name_1.GetValue(),
2940                                  'line2_name': self.line_name_2.GetValue(), 'y_lim': y_lim,
2941                                  'x_lim': x_lim, 'report_format': report_format,
2942                                  'title': self.title.GetValue(), 'author': self.author.GetValue(),
2943                                  'intro': self.intro.GetValue(), 'use_summary': self.use_summary.GetValue(),
2944                                  'summary': self.summary.GetValue(), 'dpi': dpi_value})
2945        self.parent.on_compare()
2946        self.Close()
2947
2948    def on_check(self, _):
2949        if self.use_summary.GetValue():
2950            self.summary.Enable()
2951        else:
2952            self.summary.Disable()
2953
2954
2955 class NewLineWindow(wx.Frame):
2956     def __init__(self, parent):
2957         super(NewLineWindow, self).__init__(parent, title='Add/edit manual lines', size=(500, 450))
2958         self.parent = parent
2959
2960         icon = wx.Icon('graphics/T_logo.ico', wx.BITMAP_TYPE_ICO)
2961         self.SetIcon(icon)
2962
2963         sigma_values = self.parent.get_sigma_values()
2964         self.manual_lines = self.parent.get_manual_lines()
2965
2966         if sigma_values:
2967             default_min = np.min(sigma_values)
2968             default_max = np.max(sigma_values)
2969         else:
2970             default_min = 100
2971             default_max = 800
2972
2973         panel = wx.Panel(self, -1)
2974         vsizer = wx.BoxSizer(wx.VERTICAL)
2975         vsizer2 = wx.BoxSizer(wx.VERTICAL)
2976         hsizer_outer = wx.BoxSizer(wx.HORIZONTAL)
2977         vsizer_outer = wx.BoxSizer(wx.VERTICAL)
2978
2979         font = wx.Font(12, wx.DECORATIVE, wx.NORMAL, wx.BOLD)
2980
2981         header_text = 'Add S-N line'
2982         header = wx.StaticText(panel, -1, header_text)
2983         hsizer_1 = wx.BoxSizer(wx.HORIZONTAL)
2984         header.SetFont(font)
2985         hsizer_1.Add(header)
2986         hsizer_1_2 = wx.BoxSizer(wx.HORIZONTAL)
2987         info = wx.StaticText(panel, 0, 'N = c * sigma^-m')
2988         hsizer_1_2.Add(info)
2989         vsizer.Add(hsizer_1, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2990         vsizer.Add((-1, 20))
2991         vsizer.Add(hsizer_1_2, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
2992
```

```
2993        hsizer_name = wx.BoxSizer(wx.HORIZONTAL)
2994        name_info = wx.StaticText(panel, 0, 'name:')
2995        self.name = wx.TextCtrl(panel, value='reg')
2996        hsizer_name.Add(name_info, 1)
2997        hsizer_name.Add(self.name, 2)
2998
2999        hsizer_3 = wx.BoxSizer(wx.HORIZONTAL)
3000        c = wx.StaticText(panel, 0, 'c:')
3001        self.c = wx.TextCtrl(panel, value='0.00e+00')
3002        hsizer_3.Add(c, 1)
3003        hsizer_3.Add(self.c, 2)
3004
3005        hsizer_2 = wx.BoxSizer(wx.HORIZONTAL)
3006        m = wx.StaticText(panel, 0, 'm:')
3007        self.m = wx.TextCtrl(panel, value='0.00')
3008        hsizer_2.Add(m, 1)
3009        hsizer_2.Add(self.m, 2)
3010
3011        hsizer_4 = wx.BoxSizer(wx.HORIZONTAL)
3012        min_stress = wx.StaticText(panel, 0, 'Min stress:')
3013        self.min = wx.TextCtrl(panel, value=str(default_min))
3014        hsizer_4.Add(min_stress, 1)
3015        hsizer_4.Add(self.min, 1)
3016
3017        hsizer_5 = wx.BoxSizer(wx.HORIZONTAL)
3018        max_stress = wx.StaticText(panel, 0, 'Max stress:')
3019        self.max = wx.TextCtrl(panel, value=str(default_max))
3020        hsizer_5.Add(max_stress, 1)
3021        hsizer_5.Add(self.max, 1)
3022
3023        vsizer.Add((-1, 20))
3024        vsizer.Add(hsizer_name, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
3025        vsizer.Add((-1, 10))
3026        vsizer.Add(hsizer_3, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
3027        vsizer.Add((-1, 10))
3028        vsizer.Add(hsizer_2, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
3029        vsizer.Add((-1, 10))
3030        vsizer.Add(hsizer_4, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
3031        vsizer.Add((-1, 10))
3032        vsizer.Add(hsizer_5, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
3033        vsizer.Add((-1, 20), -1)
3034
3035        add_btn = wx.Button(panel, wx.ID_ADD, label='Add', size=(100, 40))
3036        self.Bind(wx.EVT_BUTTON, self.on_add, add_btn)
3037        vsizer.Add(add_btn, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
3038
3039        view_text = 'View/edit lines'
3040        view = wx.StaticText(panel, -1, view_text)
3041        view.SetFont(font)
3042        vsizer2.Add(view)
3043
3044        self.line_info = wx.StaticText(panel, -1, '')
3045        vsizer2.Add(self.line_info, -1, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
3046
3047        hsizer_n = wx.BoxSizer(wx.HORIZONTAL)
3048        n = wx.StaticText(panel, 0, 'Line number for remove/edit: ')
3049        self.n = wx.TextCtrl(panel, value='1')
3050        hsizer_n.Add(n, 3)
3051        hsizer_n.Add(self.n, 1)
3052        vsizer2.Add(hsizer_n, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
3053
3054        edit_btn = wx.Button(panel, wx.ID_EDIT, label='Edit/Remove', size=(100, 40))
3055        self.Bind(wx.EVT_BUTTON, self.on_edit, edit_btn)
3056        vsizer2.Add(edit_btn, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
3057
3058        hsizer_outer.Add(vsizer, 1, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP | wx.BOTTOM, border=10)
3059        hsizer_outer.Add(vsizer2, 2, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP | wx.BOTTOM, border=10)
3060        vsizer_outer.Add(hsizer_outer, -1)
3061
3062        done_btn = wx.Button(panel, wx.ID_CLOSE, label='Done', size=(100, 40))
3063        self.Bind(wx.EVT_BUTTON, self.on_done, done_btn)
3064
3065        vsizer_outer.Add(done_btn, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP | wx.BOTTOM)
3066        panel.SetSizer(vsizer_outer)
3067
3068        self.update_info()
3069        self.Centre()
3070        self.Show()
3071
3072    def on_add(self, _):
3073        name = self.name.GetValue()
3074        c = self.c.GetValue()
3075        m = self.m.GetValue()
3076        max_sigma = self.max.GetValue()
3077        min_sigma = self.min.GetValue()
3078
3079        values = [c, m, min_sigma, max_sigma]
3080        for i, value in enumerate(values):
```

```
3081            if ',' in value:
3082                values[i] = value.replace(',', '.')
3083        if float(values[0]) > 0:
3084            line = ManualLine(name, eval(values[0]), eval(values[1]), eval(values[2]), eval(values[3]))
3085            self.manual_lines.append(line)
3086            self.update_info()
3087
3088    def update_info(self):
3089        line_info = ''
3090        for i, line in enumerate(self.manual_lines):
3091            line_info += 'Line %d: \n' % (i+1)
3092            line_formula, name = line.get_line_info_wx()
3093            line_info += line_formula % name
3094            line_info += '\n\n'
3095        self.line_info.SetLabel(line_info)
3096        self.Layout()
3097
3098    def on_edit(self, _):
3099        try:
3100            line_num = int(self.n.GetValue())
3101            if line_num <= len(self.manual_lines):
3102                line = self.manual_lines[line_num-1]
3103                self.name.SetValue(line.name)
3104                self.c.SetValue(str(line.c))
3105                self.m.SetValue(str(line.m))
3106                self.min.SetValue(str(line.min_sigma))
3107                self.max.SetValue(str(line.max_sigma))
3108                self.manual_lines.pop(line_num-1)
3109                self.update_info()
3110        except:
3111            pass
3112
3113    def on_done(self, _):
3114        self.parent.set_manual_lines(self.manual_lines)
3115        self.parent.on_plot()
3116        self.Destroy()
3117
3118
3119 class PlotWindow(wx.Frame):
3120     def __init__(self, parent):
3121         super(PlotWindow, self).__init__(parent, title='Plot settings', size=(300, 500))
3122         self.parent = parent
3123         self.x_lim = self.parent.get_x_lim()
3124         self.y_lim = self.parent.get_y_lim()
3125
3126         panel = wx.Panel(self, -1)
3127         vsizer = wx.BoxSizer(wx.VERTICAL)
3128         vsizer_outer = wx.BoxSizer(wx.VERTICAL)
3129
3130         icon = wx.Icon('graphics/T_logo.ico', wx.BITMAP_TYPE_ICO)
3131         self.SetIcon(icon)
3132
3133         done_btn = wx.Button(panel, wx.ID_CLOSE, label='Done', size=(100, 40))
3134
3135         font = wx.Font(12, wx.DECORATIVE, wx.NORMAL, wx.BOLD)
3136
3137         hsizer_grid = wx.BoxSizer(wx.HORIZONTAL)
3138         grid_header = wx.StaticText(panel, -1, 'Grid')
3139         grid_header.SetFont(font)
3140         hsizer_grid.Add(grid_header)
3141         vsizer.Add(hsizer_grid, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
3142
3143         hsizer_grid_settings = wx.BoxSizer(wx.HORIZONTAL)
3144         self.show_grid = wx.CheckBox(panel, label='Show grid')
3145         self.show_grid.SetValue(parent.get_show_grid())
3146         hsizer_grid_settings.Add(self.show_grid)
3147
3148         vsizer.Add((-1, 10))
3149         vsizer.Add(hsizer_grid_settings, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
3150         vsizer.Add((-1, 20))
3151
3152         x_header_text = 'X-limits'
3153         hsizer_x1 = wx.BoxSizer(wx.HORIZONTAL)
3154         x_header = wx.StaticText(panel, -1, x_header_text)
3155         x_header.SetFont(font)
3156         hsizer_x1.Add(x_header)
3157         vsizer.Add(hsizer_x1, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
3158
3159         hsizer_x2 = wx.BoxSizer(wx.HORIZONTAL)
3160         xmin = wx.StaticText(panel, 0, 'x_min:')
3161         x_min_exp = '%.2e' % (self.x_lim[0])
3162         self.x_min = wx.TextCtrl(panel, value=x_min_exp)
3163         hsizer_x2.Add(xmin, 1)
3164         hsizer_x2.Add(self.x_min, 2)
3165
3166         hsizer_x3 = wx.BoxSizer(wx.HORIZONTAL)
3167         xmax = wx.StaticText(panel, 0, 'x_max:')
3168         x_max_exp = '%.2e' % (self.x_lim[1])
```

```
3169        self.x_max = wx.TextCtrl(panel, value=x_max_exp)
3170        hsizer_x3.Add(xmax, 1)
3171        hsizer_x3.Add(self.x_max, 2)
3172
3173        vsizer.Add(hsizer_x2, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
3174        vsizer.Add((-1, 10))
3175        vsizer.Add(hsizer_x3, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
3176        vsizer.Add((-1, 20))
3177
3178        hsizer_x4 = wx.BoxSizer(wx.HORIZONTAL)
3179        self.show_xticks = wx.CheckBox(panel, label='Show minor x-ticks')
3180        if self.parent.get_xticks():
3181            self.show_xticks.SetValue(True)
3182        else:
3183            self.show_xticks.SetValue(False)
3184        hsizer_x4.Add(self.show_xticks)
3185        vsizer.Add(hsizer_x4, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
3186        vsizer.Add((-1, 30))
3187
3188        y_header_text = 'Y-limits'
3189        hsizer_y1 = wx.BoxSizer(wx.HORIZONTAL)
3190        y_header = wx.StaticText(panel, -1, y_header_text)
3191        y_header.SetFont(font)
3192        hsizer_y1.Add(y_header)
3193        vsizer.Add(hsizer_y1, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
3194
3195        hsizer_y2 = wx.BoxSizer(wx.HORIZONTAL)
3196        ymin = wx.StaticText(panel, 0, 'y_min:')
3197        self.y_min = wx.TextCtrl(panel, value=str(self.y_lim[0]))
3198        hsizer_y2.Add(ymin, 1)
3199        hsizer_y2.Add(self.y_min, 2)
3200
3201        hsizer_y3 = wx.BoxSizer(wx.HORIZONTAL)
3202        ymax = wx.StaticText(panel, 0, 'y_max:')
3203        self.y_max = wx.TextCtrl(panel, value=str(self.y_lim[1]))
3204        hsizer_y3.Add(ymax, 1)
3205        hsizer_y3.Add(self.y_max, 2)
3206
3207        vsizer.Add(hsizer_y2, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
3208        vsizer.Add((-1, 10))
3209        vsizer.Add(hsizer_y3, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
3210        vsizer.Add((-1, 20))
3211
3212        hsizer_y4 = wx.BoxSizer(wx.HORIZONTAL)
3213        self.show_yticks = wx.CheckBox(panel, label='Show minor y-ticks')
3214        if self.parent.get_yticks():
3215            self.show_yticks.SetValue(True)
3216        else:
3217            self.show_yticks.SetValue(False)
3218        hsizer_y4.Add(self.show_yticks)
3219        vsizer.Add(hsizer_y4, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
3220        vsizer.Add((-1, 30))
3221
3222        self.Bind(wx.EVT_BUTTON, self.on_done, done_btn)
3223        vsizer.Add((1, 1), -1, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
3224        vsizer.Add(done_btn, 0, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP)
3225
3226        vsizer_outer.Add(vsizer, -1, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP | wx.BOTTOM, border=10)
3227        panel.SetSizer(vsizer_outer)
3228
3229        self.Centre()
3230        self.Show()
3231
3232    def on_done(self, _):
3233        values = [self.x_min.GetValue(), self.x_max.GetValue(), self.y_min.GetValue(), self.y_max.GetValue()]
3234        for i, value in enumerate(values):
3235            if ',' in value:
3236                values[i] = value.replace(',', '.')
3237
3238        if float(values[0]) == 0:
3239            values[0] = 1
3240        if float(values[2]) == 0:
3241            values[2] = 1
3242        self.x_lim = tuple([float(values[0]), float(values[1])])
3243        self.y_lim = tuple([float(values[2]), float(values[3])])
3244
3245        self.parent.set_show_grid(self.show_grid.GetValue())
3246
3247        self.parent.set_x_lim(self.x_lim)
3248        self.parent.set_y_lim(self.y_lim)
3249        if self.show_xticks.GetValue():
3250            self.parent.set_xticks(True)
3251        else:
3252            self.parent.set_xticks(False)
3253
3254        if self.show_yticks.GetValue():
3255            self.parent.set_yticks(True)
3256        else:
```

```
3257             self.parent.set_yticks(False)
3258
3259         self.parent.on_plot()
3260         self.Close()
3261
3262
3263 if __name__ == '__main__':
3264     app = wx.App()
3265     MainFrame()
3266     app.MainLoop()
3267
```

```python
# -*- coding: utf-8 -*-

__author__ = 'Toril Rygg'
__email__ = 'torilrygg@gmail.com'

import numpy as np
from scipy.stats import f, t
from scipy import linalg


# noinspection PyPep8Naming
class Line(object):
    def __init__(self):
        self.label = 'reg'
        self.use_label = False  # use label on design line names

        self.sigma_values = None
        self.n_values = None
        self.x_values = None
        self.y_values = None

        self.alpha = 0
        self.betta = 0
        self.c = 0
        self.m = 0

        self.variance = 0  # residual mean square/variance/S^2
        self.sd = 0

        self.r2 = 0
        self.is_linear = False
        self.linear_test_ok = False     # if l >= 3 and l < n

        self.alpha_CI = []
        self.betta_CI = []
        self.c_CI = []
        self.m_CI = []
        self.conf_band = []

        self.design_names = {'design_2_sd': '2\ SD', 'design_iiw': 'IIW',
                             'design_dnv_known': 'DNV\ SD', 'design_dnv_unknown': 'DNV',
                             'design_n_sd': '%.1f\ SD',
                             'iso_90_10': 'ISO\ 90\%\ 10\%',
                             'iso_95_10': 'ISO\ 95\%\ 10\%',
                             'iso_90_05': 'ISO\ 90\%\ 05\%',
                             'iso_95_05': 'ISO\ 95\%\ 05\%',
                             'iso_90_01': 'ISO\ 90\%\ 01\%',
                             'iso_95_01': 'ISO\ 95\%\ 01\%'}

        self.design_names_simple = {'design_2_sd': '2_SD', 'design_iiw': 'IIW',
                                    'design_dnv_known': 'DNV_SD', 'design_dnv_unknown': 'DNV',
                                    'design_n_sd': '%.1f_SD',
                                    'iso_90_10': 'ISO_90%_10%',
                                    'iso_95_10': 'ISO_95%_10%',
                                    'iso_90_05': 'ISO_90%_05%',
                                    'iso_95_05': 'ISO_95%_05%',
                                    'iso_90_01': 'ISO_90%_01%',
                                    'iso_95_01': 'ISO_95%_01%'}

    def set_label(self, label):
        self.label = label

    def set_use_label(self, boolean):
        self.use_label = boolean

    def update_values(self, n_values, sigma_values):
        self.n_values = n_values
        self.sigma_values = sigma_values
        self.x_values = [np.log10(sigma) for sigma in self.sigma_values]
        self.y_values = [np.log10(n) for n in self.n_values]
        self.calculate()

    def calculate(self):
        prob = 0.05
        x_av = np.average(self.x_values)
        y_av = np.average(self.y_values)
        n = len(self.x_values)

        SXX = self.get_SXX()
        SYY = self.get_SYY()
        SXY = self.get_SXY()

        self.betta = SXY / SXX
        self.alpha = y_av - self.betta * x_av

        self.c = 10 ** self.alpha
        self.m = -self.betta

```

```
        RSS = SYY - SXY ** 2 / SXX
        SSreg = SYY - RSS
        MSE = RSS / (n - 2.)
        MSreg = SSreg / 1

        self.variance = MSE
        self.sd = np.sqrt(MSE)

        # CI
        t_p = t.ppf(1 - prob / 2., n - 2)
        x_av = np.average(self.x_values)

        alpha_diff = t_p * self.sd * np.sqrt((1./n) + (x_av**2 / SXX))
        betta_diff = t_p * self.sd / SXX ** 0.5

        self.alpha_CI = [self.alpha + alpha_diff, self.alpha - alpha_diff]
        self.betta_CI = [self.betta + betta_diff, self.betta - betta_diff]
        self.alpha_CI = sorted(self.alpha_CI)
        self.betta_CI = sorted(self.betta_CI)
        self.c_CI = [10 ** self.alpha_CI[0], 10 ** self.alpha_CI[1]]
        self.m_CI = [-self.betta_CI[0], -self.betta_CI[1]]
        self.m_CI = sorted(self.m_CI)

        # control of line
        dataset = self.restructure_by_x_level(self.x_values, self.y_values)
        n = len(self.x_values)
        l = len(dataset)

        sum_top = 0
        sum_bottom = 0

        for data in dataset:
            mi = len(data)
            xi = np.average([x for x, y in data])
            yi = np.average([y for x, y in data])
            sum_top += mi * ((self.alpha + self.betta * xi) - yi) ** 2 / (l - 2.)
            for x, y in data:
                sum_bottom += (y - yi) ** 2 / (n - l)

        F = sum_top / sum_bottom
        self.is_linear = (F <= f.ppf(1 - prob, l - 2, n - l))
        if 3 <= l < n:
            self.linear_test_ok = True
        else:
            self.linear_test_ok = False

        self.r2 = SXY ** 2 / (SXX * SYY)

    @staticmethod
    def restructure_by_x_level(x_values, y_values):
        pairs = zip(x_values, y_values)
        pair_sort = sorted(pairs)
        x_level = pair_sort[0][0]
        dataset = []
        level = []
        for x, y in pair_sort:
            if x != x_level:
                dataset.append(level)
                level = []
            level.append([x, y])
            x_level = np.mean([x for x, y in level])
        dataset.append(level)
        return dataset

    def get_SXX(self):
        x_av = np.average(self.x_values)
        return sum([(x - x_av) ** 2 for x in self.x_values])

    def get_SYY(self):
        y_av = np.average(self.y_values)
        return sum([(y - y_av) ** 2 for y in self.y_values])

    def get_SXY(self):
        y_av = np.average(self.y_values)
        x_av = np.average(self.x_values)
        return sum([(x - x_av) * (y - y_av) for x, y in zip(self.x_values, self.y_values)])

    def get_x_sq_sum(self):
        return sum([x ** 2 for x in self.x_values])

    def get_estimated_y(self):
        return [(self.alpha + self.betta * x) for x in self.x_values]

    def get_residuals(self):
        return [(y - y_est) for y, y_est in zip(self.y_values, self.get_estimated_y())]

    def get_standarized_residuals(self):
        return [res / self.sd for res in self.get_residuals()]
```

```
177
178     def get_studentized_residuals(self):
179         residuals = self.get_residuals()
180
181         x_values = self.x_values
182         n = len(x_values)
183         x_matrix = np.zeros((n, 2))
184         for i in range(n):
185             x_matrix[i, 0] = 1
186             x_matrix[i, 1] = x_values[i]
187         xtx = x_matrix.T.dot(x_matrix)
188         xtx_inv = linalg.inv(xtx)
189         H = x_matrix.dot(xtx_inv.dot(x_matrix.T))
190
191         studentized_residuals = []
192         for i in range(n):
193             studentized_residuals.append(residuals[i]/np.sqrt((1-H[i, i])*self.variance))
194         return studentized_residuals
195
196     def get_norm_points(self):
197         n = len(self.get_residuals())
198         cum_prob = [(i - 0.5) / n for i in range(1, n + 1)]
199         sd_prob = [t.ppf(p, 1e10) for p in cum_prob]
200         return sd_prob
201
202     def get_linear_status(self):
203         if self.is_linear:
204             line_status = 'be'
205         else:
206             line_status = 'NOT be'
207
208         if self.linear_test_ok:
209             linear_report = 'Linear model can ' + line_status + ' assumed with 95 % confidence '
210         else:
211             linear_report = 'Could not test for linearity'
212
213         return linear_report
214
215     def get_results_for_print(self):
216         results = ''
217         results += 'Standard deviation: {:10.5f}\n'.format(self.sd)
218         results += '\n'
219
220         results += '95 % confidence limits:\n'
221         c0 = '%.2e' % self.c_CI[0]
222         c1 = '%.2e' % self.c_CI[1]
223         if 'e+' in c0:
224             c0_comp = c0.split('e+')
225         else:
226             c0_comp = c0.split('e-')
227         if 'e+' in c1:
228             c1_comp = c1.split('e+')
229         else:
230             c1_comp = c1.split('e-')
231         c0_super = self.convert_to_superscript(c0_comp[1])
232         c1_super = self.convert_to_superscript(c1_comp[1])
233
234         results += '{}*10{}    < {:^10s} <   {}*10{}\n'.format(c0_comp[0], c0_super, 'c', c1_comp[0], c1_super)
235         results += '{:^20.2f} < {:^10s} < {:^20.2}\n'.format(self.m_CI[0], 'm', self.m_CI[1])
236         results += '\n'
237
238         results += 'Statistics: \n'
239         results += u'Goodness of fit (r\N{SUPERSCRIPT TWO}): %.3f \n' % self.r2
240         results += self.get_linear_status()
241
242         return results
243
244     def get_result_dict(self, design_lines, n_sd, use_label=False, txt=False):
245         c0 = '%.2e' % self.c_CI[0]
246         c1 = '%.2e' % self.c_CI[1]
247         if 'e+' in c0:
248             c0_comp = c0.split('e+')
249         else:
250             c0_comp = c0.split('e-')
251         if 'e+' in c1:
252             c1_comp = c1.split('e+')
253         else:
254             c1_comp = c1.split('e-')
255         if not txt:
256             c0 = r'%s \cdot 10^{%s}' % (c0_comp[0], c0_comp[1])
257             c1 = r'%s \cdot 10^{%s}' % (c1_comp[0], c1_comp[1])
258         else:
259             c0 = r'%s*10^%s' % (c0_comp[0], c0_comp[1])
260             c1 = r'%s*10^%s' % (c1_comp[0], c1_comp[1])
261         c_CI = [c0, c1]
262
263         label = self.label
264         if not txt and ' ' in label:
```

```
265             label = label.replace(' ', '\ ')
266         if not txt and '_' in label:
267             label = label.replace('_', '\_')
268 
269         if not txt:
270             reg_line, name = self.get_line_info('reg_line')
271         else:
272             reg_line, name = self.get_line_info_plain('reg_line')
273 
274         content_dict = {'sd': self.sd,
275                         'r2': self.r2,
276                         'reg_line': reg_line % label,
277                         'c_CI_low': c_CI[0],
278                         'c_CI_high': c_CI[1],
279                         'm_CI_low': self.m_CI[0],
280                         'm_CI_high': self.m_CI[1],
281                         'linear_report': self.get_linear_status()}
282 
283         for i, line in enumerate(design_lines):
284             line_key = 'design_line'+str(i)
285             if not txt:
286                 linetext, name = self.get_line_info(line, n_sd)
287             else:
288                 linetext, name = self.get_line_info_plain(line, n_sd)
289             if use_label and not txt:
290                 name = label + '\ ' + name
291             elif use_label:
292                 name = label + '_' + name
293             content_dict[line_key] = linetext % name
294 
295         return content_dict
296 
297     def get_lines_with_name(self, design_lines, n_sd, use_label=False, txt=False):
298         label = self.label
299         if not txt and ' ' in label:
300             label = label.replace(' ', '\ ')
301         if not txt and '_' in label:
302             label = label.replace('_', '\_')
303 
304         if not txt:
305             reg_line, name = self.get_line_info('reg_line')
306         else:
307             reg_line, name = self.get_line_info_plain('reg_line')
308 
309         content_dict = {'reg_line': reg_line % label}
310 
311         for i, line in enumerate(design_lines):
312             line_key = 'design_line'+str(i)
313             if not txt:
314                 linetext, name = self.get_line_info(line, n_sd)
315                 design_name = self.design_names[line]
316             else:
317                 linetext, name = self.get_line_info_plain(line, n_sd)
318                 design_name = self.design_names_simple[line]
319             if use_label and not txt:
320                 name = label + '\ ' + name
321             elif use_label:
322                 name = label + '_' + name
323             content_dict[line_key] = [linetext % name, design_name]
324         return content_dict
325 
326     def get_line_info_plain(self, linetype, n_sd=None):
327         if linetype == 'reg_line':
328             c = '%.2e' % self.c
329             if 'e+' in c:
330                 c_comp = c.split('e+')
331             else:
332                 c_comp = c.split('e-')
333             label_end = r'%s*10^%s * sigma^-%.2f' % (c_comp[0], c_comp[1], self.m)
334             label = 'N_%s = '+label_end
335             return label, 'reg'
336         else:
337             c = self.get_design_c(linetype, n_sd)
338             c_str = '%.2e' % c
339             if 'e+' in c_str:
340                 c_comp = c_str.split('e+')
341             else:
342                 c_comp = c_str.split('e-')
343             name = self.design_names_simple[linetype]
344             if 'n_sd' in linetype:
345                 name = name % n_sd
346             label_end = r'%s*10^%s * sigma^-%.2f' % (c_comp[0], c_comp[1], self.m)
347             label = 'N_%s = '+label_end
348             return label, name
349 
350     def get_line_info_wx(self, linetype, n_sd=None):
351         if linetype == 'reg_line':
352             c = '%.2e' % self.c
```

```
353             if 'e+' in c:
354                 c_comp = c.split('e+')
355             else:
356                 c_comp = c.split('e-')
357             m = '-%.2f' % self.m
358             m_super = self.convert_to_superscript(m)
359             c_super = self.convert_to_superscript(c_comp[1])
360             label_end = u'%s*10%s *\N{GREEK CAPITAL LETTER DELTA} \N{GREEK SMALL LETTER SIGMA}%s' \
361                     % (c_comp[0], c_super, m_super)
362             label = u'N_%s = '+label_end
363             return label, 'reg'
364         else:
365             c = self.get_design_c(linetype, n_sd)
366             c_str = '%.2e' % c
367             if 'e+' in c_str:
368                 c_comp = c_str.split('e+')
369             else:
370                 c_comp = c_str.split('e-')
371             m = '-%.2f' % self.m
372             m_super = self.convert_to_superscript(m)
373             c_super = self.convert_to_superscript(c_comp[1])
374             name = self.design_names_simple[linetype]
375             if 'n_sd' in linetype:
376                 name = name % n_sd
377             label_end = u'%s*10%s *\N{GREEK CAPITAL LETTER DELTA} \N{GREEK SMALL LETTER SIGMA}%s' \
378                     % (c_comp[0], c_super, m_super)
379             label = u'N_%s = '+label_end
380             return label, name
381 
382     @staticmethod
383     def convert_to_superscript(value):
384         conversions = {'1': u'\N{SUPERSCRIPT ONE}', '2': u'\N{SUPERSCRIPT TWO}', '3': u'\N{SUPERSCRIPT THREE}',
385                        '4': u'\N{SUPERSCRIPT FOUR}', '5': u'\N{SUPERSCRIPT FIVE}', '6': u'\N{SUPERSCRIPT SIX}',
386                        '7': u'\N{SUPERSCRIPT SEVEN}', '8': u'\N{SUPERSCRIPT EIGHT}', '9': u'\N{SUPERSCRIPT NINE}',
387                        '0': u'\N{SUPERSCRIPT ZERO}', '.': u'\N{APOSTROPHE}', ',': u'\N{APOSTROPHE}',
388                        '-': u'\N{SUPERSCRIPT MINUS}'}
389         new_writing = u''
390         for letter in value:
391             new_writing += conversions[letter]
392         return new_writing
393 
394     def get_line_info(self, linetype, n_sd=None):
395         if linetype == 'reg_line':
396             c = '%.2e' % self.c
397             if 'e+' in c:
398                 c_comp = c.split('e+')
399             else:
400                 c_comp = c.split('e-')
401             label_end = r'%s \cdot 10^{%s} \cdot \Delta \sigma ^{-%.2f} $' % (c_comp[0], c_comp[1], self.m)
402             label = '$\ N_{%s} = '+label_end
403             return label, 'reg'
404         else:
405             c = self.get_design_c(linetype, n_sd)
406             c_str = '%.2e' % c
407             if 'e+' in c_str:
408                 c_comp = c_str.split('e+')
409             else:
410                 c_comp = c_str.split('e-')
411             name = self.design_names[linetype]
412             if 'n_sd' in linetype:
413                 name = name % n_sd
414             label_end = r'%s \cdot 10^{%s} \cdot \Delta \sigma ^{-%.2f} $' % (c_comp[0], c_comp[1], self.m)
415             label = '$\ N_{%s} = '+label_end
416             return label, name
417 
418     def get_line_points(self, linetype, n_sd=None):  # returns values for plotting
419         sigma = np.array([min(self.sigma_values), max(self.sigma_values)])
420         if linetype == 'reg_line':
421             c = '%.2e' % self.c
422             if 'e+' in c:
423                 c_comp = c.split('e+')
424             else:
425                 c_comp = c.split('e-')
426             label_end = r'%s \cdot 10^{%s} \cdot \Delta \sigma ^{-%.2f} $' % (c_comp[0], c_comp[1], self.m)
427             label = '$\ N_{%s} = '+label_end
428             name = 'reg_line'
429             n = self.c * sigma ** float(-self.m)
430         else:
431             c = self.get_design_c(linetype, n_sd)
432             c_str = '%.2e' % c
433             if 'e+' in c_str:
434                 c_comp = c_str.split('e+')
435             else:
436                 c_comp = c_str.split('e-')
437             name = self.design_names[linetype]
438             if 'n_sd' in linetype:
439                 name = name % n_sd
440             label_end = r'%s \cdot 10^{%s} \cdot \Delta \sigma ^{-%.2f} $' % (c_comp[0], c_comp[1], self.m)
```

```
            label = '$\ N_{%s} = '+label_end

            n = c * sigma ** float(-self.m)
        return n, sigma, label, name

    def get_design_c(self, designtype, n_sd=None):
        n = len(self.n_values)
        if 'dnv' in designtype:
            if 'unknown' in designtype:
                factor = self.get_dnv_factor(False)
            else:
                factor = self.get_dnv_factor(True)
        elif 'iso' in designtype:
            factor = self.get_ISO_factor(designtype[4:])
        elif 'iiw' in designtype:
            factor = np.sqrt((n+1.)/n)*t.ppf(1 - 0.05 / 2, n-2)
        elif 'n_sd' in designtype:
            factor = n_sd
        else:
            factor = 2

        log_c_d = np.log10(self.c) - factor * self.sd
        c_design = 10 ** log_c_d
        return c_design

    def get_dnv_factor(self, sd_known=True):
        if sd_known:
            usecol = 1
        else:
            usecol = 2

        factors = [[3, 2.39, 9.24],
                   [5, 2.30, 5.01],
                   [10, 2.21, 3.45],
                   [15, 2.17, 3.07],
                   [20, 2.15, 2.88],
                   [30, 2.12, 2.65],
                   [50, 2.10, 2.48],
                   [100, 2.07, 2.32],
                   [1e100, 2.00, 2.00]]

        n = len(self.n_values)
        factor = None

        for line in factors:
            if n > line[0]:
                factor = line[usecol]
            else:
                return factor

    def get_ISO_factor(self, iso_type='90_05'):
        n = len(self.n_values)
        k_factors = {'90_10': {'2':    4.258,
                               '3':    3.187,
                               '4':    2.742,
                               '5':    2.494,
                               '6':    2.333,
                               '7':    2.219,
                               '8':    2.133,
                               '9':    2.065,
                               '10':   2.012,
                               '11':   1.966,
                               '12':   1.928,
                               '13':   1.895,
                               '14':   1.866,
                               '15':   1.842,
                               '16':   1.820,
                               '17':   1.800,
                               '18':   1.781,
                               '19':   1.765,
                               '20':   1.750,
                               '21':   1.736,
                               '22':   1.724,
                               '23':   1.712,
                               '24':   1.702,
                               '25':   1.657},
                     '95_10': {'2':    6.158,
                               '3':    4.163,
                               '4':    3.407,
                               '5':    3.006,
                               '6':    2.755,
                               '7':    2.582,
                               '8':    2.454,
                               '9':    2.355,
                               '10':   2.275,
                               '11':   2.210,
                               '12':   2.155,
                               '13':   2.108,
```

```
                '14':  2.068,
                '15':  2.032,
                '16':  2.001,
                '17':  1.974,
                '18':  1.949,
                '19':  1.926,
                '20':  1.905,
                '21':  1.887,
                '22':  1.869,
                '23':  1.853,
                '24':  1.838,
                '25':  1.778},
            '90_05': {'2':   5.310,
                '3':   3.957,
                '4':   3.400,
                '5':   3.091,
                '6':   2.894,
                '7':   2.755,
                '8':   2.649,
                '9':   2.568,
                '10':  2.503,
                '11':  2.448,
                '12':  2.403,
                '13':  2.363,
                '14':  2.329,
                '15':  2.299,
                '16':  2.272,
                '17':  2.249,
                '18':  2.228,
                '19':  2.208,
                '20':  2.190,
                '21':  2.174,
                '22':  2.159,
                '23':  2.145,
                '24':  2.132,
                '25':  2.080},
            '95_05': {'2':   7.655,
                '3':   5.145,
                '4':   4.202,
                '5':   3.707,
                '6':   3.399,
                '7':   3.188,
                '8':   3.031,
                '9':   2.911,
                '10':  2.815,
                '11':  2.736,
                '12':  2.670,
                '13':  2.614,
                '14':  2.566,
                '15':  2.523,
                '16':  2.486,
                '17':  2.453,
                '18':  2.423,
                '19':  2.396,
                '20':  2.371,
                '21':  2.350,
                '22':  2.329,
                '23':  2.309,
                '24':  2.292,
                '25':  2.220},
            '90_01': {'2':   7.340,
                '3':   5.437,
                '4':   4.666,
                '5':   4.242,
                '6':   3.972,
                '7':   3.783,
                '8':   3.641,
                '9':   3.532,
                '10':  3.444,
                '11':  3.370,
                '12':  3.310,
                '13':  3.257,
                '14':  3.212,
                '15':  3.172,
                '16':  3.136,
                '17':  3.106,
                '18':  3.078,
                '19':  3.052,
                '20':  3.028,
                '21':  3.007,
                '22':  2.987,
                '23':  2.969,
                '24':  2.952,
                '25':  2.884},
            '95_01': {'2':   10.55,
                '3':   7.042,
                '4':   5.741,
                '5':   5.062,
```

```
                    '6':  4.641,
                    '7':  4.353,
                    '8':  4.143,
                    '9':  3.981,
                    '10':  3.852,
                    '11':  3.747,
                    '12':  3.659,
                    '13':  3.585,
                    '14':  3.520,
                    '15':  3.463,
                    '16':  3.415,
                    '17':  3.370,
                    '18':  3.331,
                    '19':  3.295,
                    '20':  3.262,
                    '21':  3.233,
                    '22':  3.206,
                    '23':  3.181,
                    '24':  3.158,
                    '25':  3.064}
                }

        if n <= 27:
            k = k_factors[iso_type][str(n-2)]
        else:
            k = k_factors[iso_type]['25']
        return np.sqrt((n+1.)/n)*k

    def get_ci_points(self):
        x_values1 = np.linspace(np.min(self.x_values), np.max(self.x_values), 20)
        y_values1 = [self.get_ci_at_x(x)[0] for x in x_values1]
        y_values2 = [self.get_ci_at_x(x)[1] for x in x_values1]

        sigma = [10 ** x for x in x_values1]
        n1 = [10 ** y for y in y_values1]
        n2 = [10 ** y for y in y_values2]

        return n1, n2, sigma

    def get_ci_at_x(self, x_place):
        n = self.get_num_values()
        prob = 0.05
        fp = f.ppf(1 - prob, 2, n - 2)
        x_av = np.mean(self.x_values)
        sxx = sum([(x - x_av) ** 2 for x in self.x_values])

        alpha = self.alpha
        betta = self.betta
        diff = np.sqrt(2*fp) * self.sd * np.sqrt((1 / n) + ((x_place - x_av)**2 / sxx))

        l1 = alpha + betta * x_place + diff
        l2 = alpha + betta * x_place - diff
        return l1, l2

    def get_ellipse_points(self):
        betta_CI = sorted(self.betta_CI)
        betta_values1 = np.linspace(betta_CI[0] - 0.2, betta_CI[1] + 0.2, 50)
        alpha_values1 = [self.get_alpha_at_betta(betta)[0] for betta in betta_values1]
        alpha_values2 = [self.get_alpha_at_betta(betta)[1] for betta in betta_values1]
        alpha_values1.extend(alpha_values2[::-1])
        betta_values1 = list(betta_values1)
        betta_values1.extend(betta_values1[::-1])
        alpha_values = []
        betta_values = []
        for alpha, betta in zip(alpha_values1, betta_values1):
            if not np.isnan(alpha):
                alpha_values.append(alpha)
                betta_values.append(betta)
        betta_values.append(betta_values[0])
        alpha_values.append(alpha_values[0])

        c_values = [10**a for a in alpha_values]
        m_values = [-b for b in betta_values]

        return c_values, m_values

    def get_CI_lines(self):
        c_low = min(self.c_CI)
        c_high = max(self.c_CI)
        m_low = min(self.m_CI)
        m_high = max(self.m_CI)

        lines = [[[c_low, c_low], [m_low-2, m_high+2]],
                 [[c_high, c_high], [m_low-2, m_high+2]],
                 [[c_low/4., c_high*4.], [m_low, m_low]],
                 [[c_low/4., c_high*4.], [m_high, m_high]]]
        return lines

```

```
    def get_alpha_at_betta(self, betta):
        n = self.get_num_values()
        prob = 0.05
        fp = f.ppf(1 - prob, 2, n - 2)

        sum_x = sum(self.x_values)
        sum_xx = sum([x ** 2 for x in self.x_values])

        A = n
        B = (-2) * n * self.alpha - 2 * sum_x * self.betta + 2 * sum_x * betta
        C = n * self.alpha ** 2 + 2 * sum_x * self.alpha * self.betta - 2 * sum_x * self.alpha * betta \
            + sum_xx * (self.betta - betta) ** 2 - 2 * fp * self.sd ** 2

        root = B ** 2 - 4 * A * C
        if root >= 0:
            diff = np.sqrt(B ** 2 - 4 * A * C)
            alpha1 = (-B + diff) / (2 * A)
            alpha2 = (-B - diff) / (2 * A)
        else:
            alpha1 = np.nan
            alpha2 = np.nan
        return alpha1, alpha2

    def get_c(self):
        return self.c

    def get_m(self):
        return self.m

    def get_c_CI(self):
        return self.c_CI

    def get_m_CI(self):
        return self.m_CI

    def get_variance(self):
        return self.variance

    def get_sd(self):
        return self.sd

    def get_num_values(self):
        if self.x_values:
            return len(self.x_values)
        else:
            return 0

    def get_alpha(self):
        return self.alpha

    def get_betta(self):
        return self.betta

    def get_data(self):
        return self.sigma_values, self.n_values

    def get_x_values(self):
        return self.x_values

    def get_y_values(self):
        return self.y_values


class ManualLine(object):
    def __init__(self, name, c, m, min_sigma, max_sigma):
        self.name = name
        self.c = c
        self.m = m
        self.min_sigma = min_sigma
        self.max_sigma = max_sigma
        self.sigma_values = np.array([min_sigma, max_sigma])
        self.n_values = c * self.sigma_values ** float(-m)

    def get_line_info_plain(self):
        c = '%.2e' % self.c
        if 'e+' in c:
            c_comp = c.split('e+')
        else:
            c_comp = c.split('e-')
        label_end = '%s*10^%s * sigma^-%.2f' % (c_comp[0], c_comp[1], self.m)
        label = 'N_%s =' + label_end
        return label, self.name

    def get_line_info_wx(self):
        c = '%.2e' % self.c
        if 'e+' in c:
            c_comp = c.split('e+')
        else:
```

```
            c_comp = c.split('e-')
        m = '-%.2f' % self.m
        m_super = self.convert_to_superscript(m)
        c_super = self.convert_to_superscript(c_comp[1])
        label_end = u'%s*10%s *\N{GREEK CAPITAL LETTER DELTA} \N{GREEK SMALL LETTER SIGMA}%s' \
                    % (c_comp[0], c_super, m_super)
        label = u'N_%s = '+label_end
        return label, self.name

    @staticmethod
    def convert_to_superscript(value):
        conversions = {'1': u'\N{SUPERSCRIPT ONE}', '2': u'\N{SUPERSCRIPT TWO}', '3': u'\N{SUPERSCRIPT THREE}',
                       '4': u'\N{SUPERSCRIPT FOUR}', '5': u'\N{SUPERSCRIPT FIVE}', '6': u'\N{SUPERSCRIPT SIX}',
                       '7': u'\N{SUPERSCRIPT SEVEN}', '8': u'\N{SUPERSCRIPT EIGHT}', '9': u'\N{SUPERSCRIPT NINE}',
                       '0': u'\N{SUPERSCRIPT ZERO}', '.': u'\N{APOSTROPHE}', ',': u'\N{APOSTROPHE}',
                       '-': u'\N{SUPERSCRIPT MINUS}'}
        new_writing = u''
        for letter in value:
            new_writing += conversions[letter]
        return new_writing

    def get_line_info(self):
        c_str = '%.2e' % self.c
        if 'e+' in c_str:
            c_comp = c_str.split('e+')
        else:
            c_comp = c_str.split('e-')
        label_end = '%s \cdot 10^{%s} \cdot \Delta \sigma ^{-%.2f} $' % (c_comp[0], c_comp[1], self.m)
        label = r'$ N_{%s} =' + label_end
        return label, self.name

    def get_line_points(self):  # returns values for plotting
        label, name = self.get_line_info()
        return self.n_values, self.sigma_values, label, self.name


class CompareLines(object):
    def __init__(self, line1, line2):
        self.line1 = line1
        self.line2 = line2

        self.prob = 0.5

    def compare_variance(self, double_space=False):
        fp = f.ppf(1 - self.prob / 2., self.line1.get_num_values() - 2, self.line2.get_num_values() - 2)

        try:
            div_var = self.line1.get_sd() ** 2 / self.line2.get_sd() ** 2

            if div_var > fp or div_var < 1. / fp:
                return 'Variances can NOT be assumed equal with 95 % confidence.'
            else:
                variance = self.get_av_variance()
                report = 'Variances can be assumed equal with 95 % confidence. \n'
                if double_space:
                    report += '\n'
                report += 'Common variance is %.6f. (SD is %.4f)' % (variance, float(np.sqrt(variance)))
                return report
        except:
            return 'Could not test for variance'

    def get_av_variance(self):
        n1 = self.line1.get_num_values()
        n2 = self.line2.get_num_values()
        var1 = self.line1.get_variance()
        var2 = self.line2.get_variance()
        return ((n1-2)*var1 + (n2-2)*var2)/float(n1+n2+4)

    @staticmethod
    def get_full_matrix(x_values, n1, n2):
        n_tot = n1 + n2
        x_matrix = np.zeros((n_tot, 4))
        for n in range(n_tot):
            if n < n1:
                alpha_col = 0
                betta_col = 2
            else:
                alpha_col = 1
                betta_col = 3
            x_matrix[n, alpha_col] = 1
            x_matrix[n, betta_col] = x_values[n]
        return x_matrix

    @staticmethod
    def get_simple_matrix(x_values, n1, n2):
        n_tot = n1 + n2
        x_matrix = np.zeros((n_tot, 2))
        for n in range(n_tot):
```

```
            x_matrix[n, 0] = 1
            x_matrix[n, 1] = x_values[n]
        return x_matrix

    @staticmethod
    def get_alpha_matrix(x_values, n1, n2):
        n_tot = n1 + n2
        x_matrix = np.zeros((n_tot, 3))
        for n in range(n_tot):
            if n < n1:
                betta_col = 1
            else:
                betta_col = 2
            x_matrix[n, 0] = 1
            x_matrix[n, betta_col] = x_values[n]
        return x_matrix

    @staticmethod
    def get_betta_matrix(x_values, n1, n2):
        n_tot = n1 + n2
        x_matrix = np.zeros((n_tot, 3))
        for n in range(n_tot):
            if n < n1:
                alpha_col = 0
            else:
                alpha_col = 1
            x_matrix[n, alpha_col] = 1
            x_matrix[n, 2] = x_values[n]
        return x_matrix

    @staticmethod
    def get_SSE(x_matrix, y_matrix):
        xtx = x_matrix.T.dot(x_matrix)
        xtx_inv = linalg.inv(xtx)
        H = x_matrix.dot(xtx_inv.dot(x_matrix.T))
        I = np.identity(len(y_matrix))
        e = (I-H).dot(y_matrix)
        SSE = e.T.dot(e)
        return SSE

    def compare_slope(self):
        x1 = self.line1.get_x_values()
        y1 = self.line1.get_y_values()
        x2 = self.line2.get_x_values()
        y2 = self.line2.get_y_values()
        x_values = x1 + x2
        n1 = len(x1)
        n2 = len(x2)

        y_matrix = np.array(y1 + y2)
        x_matrix_full = self.get_full_matrix(x_values, n1, n2)
        SSE_H1 = self.get_SSE(x_matrix_full, y_matrix)

        x_matrix_betta = self.get_betta_matrix(x_values, n1, n2)
        SSE_H0 = self.get_SSE(x_matrix_betta, y_matrix)

        F = ((SSE_H0-SSE_H1)/SSE_H1)*((n1+n2-4)/1.)
        fp = f.ppf(1 - self.prob, 1, n1+n2-4)
        if F > fp:
            return 'The lines can NOT be assumed parallell with 95 % confidence.'
        else:
            return 'The lines can be assumed parallell with 95 % confidence.'

    def compare_intercept(self):
        x1 = self.line1.get_x_values()
        y1 = self.line1.get_y_values()
        x2 = self.line2.get_x_values()
        y2 = self.line2.get_y_values()
        x_values = x1 + x2
        n1 = len(x1)
        n2 = len(x2)

        y_matrix = np.array(y1 + y2)
        x_matrix_full = self.get_full_matrix(x_values, n1, n2)
        SSE_H1 = self.get_SSE(x_matrix_full, y_matrix)

        x_matrix_alpha = self.get_alpha_matrix(x_values, n1, n2)
        SSE_H0 = self.get_SSE(x_matrix_alpha, y_matrix)

        F = ((SSE_H0-SSE_H1)/SSE_H1)*((n1+n2-4)/1.)
        fp = f.ppf(1 - self.prob, 1, n1+n2-4)
        if F > fp:
            return 'The intercepts can NOT be assumed equal with 95 % confidence.'
        else:
            return 'The intercepts can be assumed equal with 95 % confidence.'

    def compare_lines(self):
        x1 = self.line1.get_x_values()
```

```
        y1 = self.line1.get_y_values()
        x2 = self.line2.get_x_values()
        y2 = self.line2.get_y_values()
        x_values = x1 + x2
        n1 = len(x1)
        n2 = len(x2)

        y_matrix = np.array(y1 + y2)
        x_matrix_full = self.get_full_matrix(x_values, n1, n2)
        SSE_H1 = self.get_SSE(x_matrix_full, y_matrix)

        x_matrix_simple = self.get_simple_matrix(x_values, n1, n2)
        SSE_H0 = self.get_SSE(x_matrix_simple, y_matrix)

        F = ((SSE_H0-SSE_H1)/SSE_H1)*((n1+n2-4)/2.)
        fp = f.ppf(1 - self.prob, 2, n1+n2-4)
        if F > fp:
            return 'The lines can NOT be assumed coincident with 95 % confidence.'
        else:
            return 'The lines can be assumed coincident with 95 % confidence.'

    def get_results(self):
        results = ''
        results += self.compare_variance()
        results += '\n'
        results += self.compare_slope()
        results += '\n'
        results += self.compare_intercept()
        results += '\n'
        results += self.compare_lines()
        results += '\n'
        return results

    def get_results_double_lines(self):
        results = ''
        results += self.compare_variance(double_space=True)
        results += '\n\n'
        results += self.compare_slope()
        results += '\n\n'
        results += self.compare_intercept()
        results += '\n\n'
        results += self.compare_lines()
        results += '\n\n'
        return results
```

```
# -*- coding: utf-8 -*-

__author__ = 'Toril Rygg'
__email__ = 'torilrygg@gmail.com'

import subprocess


class ReportGenerator(object):
    def __init__(self, folder_path=None, report_name='report'):
        self.folder_path = folder_path
        self.template_file = None
        self.report_name = report_name

    def make_template(self, num_data, num_design_lines, num_manual_lines, summary=False):
        file_format = '.tex'
        report_name = self.report_name

        self.template_file = self.folder_path + '\\' + report_name + file_format

        template = r"""\documentclass[12pt,a4paper]{article}
\usepackage[margin=20mm]{geometry}
\usepackage{graphicx}
\usepackage{parskip}
\usepackage{fancyhdr}
\pagestyle{fancy}
\usepackage[utf8]{inputenc}

\fancypagestyle{firstpage}
{
   \fancyhf{}
   \fancyfoot[R]{\thepage}
   \renewcommand{\headrulewidth}{0pt}
}

\fancyhead{}
\fancyfoot{}

\fancyhead[L]{Toril: Total Overview of fatigue Resistance with Increasing Loads}
\fancyhead[R]{NMBU}
\fancyfoot[R]{\thepage}

\begin{document}
\title{%(title)s}
\author{%(author)s}
\date{\today}
\maketitle
\thispagestyle{firstpage}
\renewcommand{\arraystretch}{1.5}

%(intro)s

\section{Input}
\begin{tabular}{|l|l|}
\hline
$ \Delta \sigma $ & Number \\
\hline
"""
        for i in range(num_data):
            s_key = 's'+str(i)
            n_key = 'n'+str(i)
            template += '%('+s_key+')s & %('+n_key+')s'
            template += r"""\\
"""

        template += r"""\hline
\end{tabular}

\newpage
\section{Output}

S-N line: %(reg_line)s

"""

        for i in range(num_design_lines):
            line_key = 'design_line'+str(i)
            template += '%('+line_key+')s'
            template += r"""\\
"""
        for i in range(num_manual_lines):
            line_key = 'manual_line'+str(i)
            template += '%('+line_key+')s'
            template += r"""\\
"""

        template += r"""
Standard deviation: %(sd).3f
```

```
89
90  95 %(percent)s confidence limits for m and c:
91  \begin{eqnarray}
92  %(c_CI_low)s & < c < & %(c_CI_high)s \nonumber \\
93  %(m_CI_low).2f & < m < & %(m_CI_high).2f \nonumber
94  \end{eqnarray}
95
96  \textbf{Simultaneous confidence interval:}
97
98  \includegraphics[width=150mm]{%(CI_plot)s}
99
100
101 \newpage
102 \subsection{S-N plot}
103
104 \includegraphics[width=\textwidth]{%(sn_plot)s}
105
106 \newpage
107 \subsection{Statistical analysis}
108
109 Goodness of fit ($ r^2 $): %(r2).3f
110
111 Adequacy of a linear model: %(linear_report)s \\
112
113 \textbf{Residual plot:}
114
115 \includegraphics[width=110mm]{%(residual_plot)s}
116
117 \textbf{Normal probability plot:}
118
119 \includegraphics[width=110mm]{%(norm_prob_plot)s}
120 '''
121
122         if summary:
123             template += r'''
124 \newpage
125 \subsection{Summary}
126
127 %(summary)s
128
129 '''
130
131         template += r'''
132 \end{document} '''
133
134         with open(self.template_file, 'w') as infile:
135             infile.write(template)
136
137     def create_document(self, content_dict):
138         template = file(self.template_file, 'r').read()
139         new_temp = template % content_dict
140         file(self.template_file, 'w').write(new_temp)
141
142         latex_call = 'pdflatex %s' % self.template_file
143
144         latex_call = r'MikTeX\miktex\bin\pdflatex %s' % self.template_file
145
146         subprocess.call(latex_call, cwd=self.folder_path)
147
148 class ReportGeneratorTXT(object):
149     def __init__(self, folder_path=None, report_name='report'):
150         self.folder_path = folder_path
151         self.template = None
152         self.report_name = report_name
153
154     def make_template(self, num_data, num_design_lines, num_manual_lines, summary=False):
155         template = r'''**************************************
156 %(title)s
157 %(author)s
158 %(date)s
159 **************************************
160
161 %(intro)s
162
163 *** INPUT ***
164
165 ----------------------------------------
166 Sigma            Number
167 ----------------------------------------
168 '''
169         for i in range(num_data):
170             s_key = 's'+str(i)
171             n_key = 'n'+str(i)
172             template += '%('+s_key+')10s %('+n_key+')15s'
173             template += r'''
174 '''
175
176         template += r'''
```

```
177 ---------------------------------------
178
179 *** OUTPUT ***
180
181 S-N line: %(reg_line)s
182
183 '''
184         for i in range(num_design_lines):
185             line_key = 'design_line'+str(i)
186             template += '%('+line_key+')s'
187             template += r'''
188 '''
189         for i in range(num_manual_lines):
190             line_key = 'manual_line'+str(i)
191             template += '%('+line_key+')s'
192             template += r'''
193 '''
194
195         template += r'''
196 Standard deviation: %(sd).3f
197
198 95 %(percent)s confidence limits for m and c:
199
200 %(c_CI_low)s   <   c   <   %(c_CI_high)s
201 %(m_CI_low).2f <   m   <   %(m_CI_high).2f
202
203
204 Statistical analysis:
205 ---------------------
206
207 Goodness of fit (r^2): %(r2).3f
208
209 Adequacy of a linear model: %(linear_report)s
210
211 '''
212
213         if summary:
214             template += r'''
215 Summary:
216 ---------
217 %(summary)s
218
219 '''
220         self.template = template
221
222     def create_document(self, content_dict):
223         file_format = '.txt'
224         report_name = self.report_name
225
226         filename = self.folder_path + '\\' + report_name + file_format
227
228         content = self.template % content_dict
229         with open(filename, 'w') as infile:
230             infile.write(content)
231
232
233 class CompareReportGenerator(object):
234     def __init__(self, folder_path=None, report_name='report'):
235         self.folder_path = folder_path
236         self.template_file = None
237         self.report_name = report_name
238
239     def make_template(self, num_data1, num_data2, num_design_lines, summary=False):
240         file_format = '.tex'
241         report_name = self.report_name
242
243         self.template_file = self.folder_path + '\\' + report_name + file_format
244
245         template = r'''\documentclass[12pt,a4paper]{article}
246 \usepackage[margin=20mm]{geometry}
247 \usepackage{graphicx}
248 \usepackage{parskip}
249 \usepackage{fancyhdr}
250 \pagestyle{fancy}
251 \usepackage[utf8]{inputenc}
252
253 \fancypagestyle{firstpage}
254 {
255     \fancyhf{}
256     \fancyfoot[R]{\thepage}
257     \renewcommand{\headrulewidth}{0pt}
258 }
259
260 \fancyhead{}
261 \fancyfoot{}
262
263 \fancyhead[L]{Toril: Total Overview of fatigue Resistance with Increasing Loads}
264 \fancyhead[R]{NMBU}
```

```
\fancyfoot[R]{\thepage}

\begin{document}
\title{%(title)s}
\author{%(author)s}
\date{\today}
\maketitle
\thispagestyle{firstpage}
\renewcommand{\arraystretch}{1.5}

%(intro)s

\section{Input}
\begin{tabular}{|l|l|l|l|}
\hline
\multicolumn{2}{|c|}{%(name1)s} & \multicolumn{2}{|c|}{%(name2)s}\\
\hline
$\Delta \sigma$ & N cycles & $\Delta \sigma$ & N cycles \\
\hline
'''
        n_max = max(num_data1, num_data2)
        for i in range(n_max):
            if i < num_data1 and i < num_data2:
                s1_key = 's1'+str(i)
                n1_key = 'n1'+str(i)
                s2_key = 's2'+str(i)
                n2_key = 'n2'+str(i)
                template += '%('+s1_key+')s & %('+n1_key+')s & %('+s2_key+')s & %('+n2_key+')s'
                template += r'''\\
'''
            elif i < num_data1:
                s1_key = 's1'+str(i)
                n1_key = 'n1'+str(i)
                template += '%('+s1_key+')s & %('+n1_key+')s &  & '
                template += r'''\\
'''
            else:
                s2_key = 's2'+str(i)
                n2_key = 'n2'+str(i)
                template += ' & & %('+s2_key+')s & %('+n2_key+')s'
                template += r'''\\
'''

        template += r'''\hline
\end{tabular}

\newpage
\section{Output}

\begin{tabular}{|l|l|l|}
\hline
Type & %(name1)s & %(name2)s \\
\hline
S-N line & %(line_1)s & %(line_2)s \\
'''
        for i in range(num_design_lines):
            line_name = 'design_line'+str(i)
            line_key1 = 'design_line'+str(i)+'_1'
            line_key2 = 'design_line'+str(i)+'_2'
            template += '%('+line_name+')s & %('+line_key1+')s & %('+line_key2+')s'
            template += r'''\\
'''

        template += r'''\hline
\end{tabular}

\subsection{Statistical analysis}

%(results)s

\subsection{Plot}

\includegraphics[width=\textwidth]{%(compare_plot)s}

'''

        if summary:
            template += r'''
\newpage
\subsection{Summary}

%(summary)s

'''

        template += r'''
\end{document} '''

```

```
353         with open(self.template_file, 'w') as infile:
354             infile.write(template)
355 
356     def create_document(self, content_dict):
357         template = file(self.template_file, 'r').read()
358         new_temp = template % content_dict
359         file(self.template_file, 'w').write(new_temp)
360 
361         latex_call = 'pdflatex %s' % self.template_file
362 
363         latex_call = r'MikTeX\miktex\bin\pdflatex %s' % self.template_file
364 
365         subprocess.call(latex_call, cwd=self.folder_path)
366 
367 class CompareReportGeneratorTXT(object):
368     def __init__(self, folder_path=None, report_name='report'):
369         self.folder_path = folder_path
370         self.template = None
371         self.report_name = report_name
372 
373     def make_template(self, num_data1, num_data2, num_design_lines, summary=False):
374         template = r"""************************************
375 %(title)s
376 %(author)s
377 %(date)s
378 ************************************
379 
380 %(intro)s
381 
382 *** INPUT ***
383 
384           %(name1)s                    %(name2)s
385 ------------------------------------------------------------------------
386   Delta sigma       N cycles    Delta sigma       N cycles
387 ------------------------------------------------------------------------
388 """
389         n_max = max(num_data1, num_data2)
390         for i in range(n_max):
391             if i < num_data1 and i < num_data2:
392                 s1_key = 's1'+str(i)
393                 n1_key = 'n1'+str(i)
394                 s2_key = 's2'+str(i)
395                 n2_key = 'n2'+str(i)
396 
397                 template += '%('+s1_key+')15s %('+n1_key+')15s %('+s2_key+')15s %('+n2_key+')15s'
398                 template += '\n'
399             elif i < num_data1:
400                 s1_key = 's1'+str(i)
401                 n1_key = 'n1'+str(i)
402                 template += '%('+s1_key+')15s %('+n1_key+')15s'
403                 template += '\n'
404             else:
405                 s2_key = 's2'+str(i)
406                 n2_key = 'n2'+str(i)
407                 template += '%(blank)15s %(blank)15s %('+s2_key+')15s %('+n2_key+')15s'
408                 template += '\n'
409 
410         template += r"""
411 ------------------------------------------------------------------------
412 
413 *** OUTPUT ***
414 
415 S-N line:
416 %(name1)s: %(line_1)s
417 %(name2)s: %(line_2)s
418 
419 """
420         for i in range(num_design_lines):
421             line_name = 'design_line'+str(i)
422             line_key1 = 'design_line'+str(i)+'_1'
423             line_key2 = 'design_line'+str(i)+'_2'
424             template += '%('+line_name+')s: \n'
425             template += '%(name1)s: %('+line_key1+')s \n'
426             template += '%(name2)s: %('+line_key2+')s \n'
427             template += '\n'
428 
429         template += r"""
430 Statistical analysis:
431 ---------------------
432 
433 %(results)s
434 """
435 
436         if summary:
437             template += r"""
438 Summary:
439 ---------
440 %(summary)s
```

```
441
442 '''
443         self.template = template
444
445     def create_document(self, content_dict):
446         file_format = '.txt'
447         report_name = self.report_name
448
449         filename = self.folder_path + '\\' + report_name + file_format
450
451         content = self.template % content_dict
452         with open(filename, 'w') as infile:
453             infile.write(content)
```

```python
# -*- coding: utf-8 -*-

__author__ = 'Toril Rygg'
__email__ = 'torilrygg@gmail.com'

import py2exe
from distutils.core import setup
from glob import glob
import sys
import matplotlib

sys.path.append('C:\dlls')

gui = {'script': 'Toril.py',
       'icon_resources': [(1, 'graphics\T_white.ico')],
}

data_files = [
    'help.txt',
    'Brukerveiledning.pdf',
    'models.py',
    'report_generator.py',
    'sample_data.csv',
    'sample_data_compare_1.csv',
    'sample_data_compare_2.csv',
    ('graphics', glob('graphics/*.*'))
]

py2exe_excludes = [
    '_gtkagg',
    '_tkagg',
    'bsddb',
    'curses',
    'email',
    'pywin.debugger',
    'pywin.debugger.dbgcon',
    'pywin.dialogs',
    'tcl',
    'Tkconstants',
    'Tkinter',
]

py2exe_includes = [
    'scipy.sparse.csgraph._validation',
    'scipy.special._ufuncs_cxx',
]

setup(
    name='Toril',
    options={
        'py2exe': {
            'excludes': py2exe_excludes,
            'includes': py2exe_includes,
            'dll_excludes': ['libzmq.pyd'],
        }
    },
    windows=[gui],
    data_files=data_files + matplotlib.get_py2exe_datafiles(),
    requires=['numpy', 'matplotlib', 'math', 'scipy', 'os', 'wx', 'subprocess']
)

```

Norges miljø- og
biovitenskapelige
universitet

Postboks 5003
NO-1432 Ås
67 23 00 00
www.nmbu.no