Importera, modifiera och uppdatera

Microsoft Excel-tillämpning för manuell framtagning av lastfallskombinationer i Robot Structural Analysis

Jack Sandén

Examensarbete för ingenjörsexamen (YH)

Utbildningsprogrammet för byggnadsteknik

Vasa 2014

EXAMENSARBETE

Författare:	Jack Sandén
Utbildningsprogram och ort:	Byggnadsteknik, Vasa
Inriktningsalternativ/Fördjupning:	Konstruktionsteknik
Handledare:	Anders Borg och Allan Andersson

Titel: *Importera, modifiera och uppdatera – Microsoft Excel-tillämpning för manuell framtagning av lastfallskombinationer i Robot Structural Analysis*

Datum 28.04.2014 Sidantal 32 Bilagor 3

Abstrakt

Syftet med föreliggande arbete är att utveckla och presentera ett funktionellt och användbart dataprogram som effektivt och tidsbesparande kan generera och modifiera lastfallskombinationer. Med lastfallskombinationer avses förenklat ofördelaktigt kombinerade belastningar på konstruktioner. Den teoretiska bakgrunden utgörs av de dimensioneringsprinciper som gäller i Finland. För att kunna identifiera möjliga förbättringsalternativ är en ingående kännedom om hur dessa kombinationer framtas nödvändig. I den teoretiska översikten redovisas hur lastfallskombinationer tas fram med hjälp av standardiserade direktiv.

Empirin i forskningen utgörs av ett utvecklingsarbete i form av ett dataprogram baserat på programmering i Visual Basic. Forskningen fokuserar på att kreativt kombinera de möjligheter som Microsoft Excel och Robot Structural Analysis erbjuder.

Resultaten visar att det är tämligen enkelt att skapa dataapplikationer som är användbara vid byggnadskonstruktion och som avsevärt kan underlätta annars tidsdryga rutinberäkningar och inmatningar.

Språk: svenska Nyckelord: lastfallskombinationer, Microsoft Excel, Robot Structural Analysis

---

OPINNÄYTETYÖ

Tekijä:	Jack Sandén
Koulutusohjelma ja paikkakunta:	Rakennustekniikka, Vaasa
Suuntautumisvaihtoehto:	Rakennesuunnittelu
Ohjaajat:	Allan Andersson ja Anders Borg

Nimike: *Tuonti, muokkaaminen ja päivittäminen – kuormayhdistelmien manuaalinen tuottaminen Robot Structural Analysis-ohjelmassa Microsoft Excel-sovelluksen avulla*

Päivämäärä 28.04.2014 Sivumäärä 32 Liitteet 3

Tiivistelmä

Lopputyön tavoitteena on ollut kehittää ja esittää toimivaa ja hyödyllistä dataohjelmaa, joka tehokkaasti ja aikaa säästävällä tavalla voi tuottaa ja muokata kuormayhdistelmiä.

Kuormayhdistelmä-käsitteellä tarkoitetaan yksinkertaistetusti rakenteissa epäedullisella tavalla yhdisteltyjä kuormituksia. Teoreettinen viitekehys koostuu Suomessa sovellettavista mitoitusperiaatteista. Mahdollisten parannusvaihtoehtojen tunnistaminen edellyttää perusteellista tuntemusta siitä, miten kyseisiä yhdistelmiä tehdään. Teoreettisessa yleiskatsauksessa kuvataan miten kuormayhdistelmiä saadaan aikaan käyttäen standardoituja direktiivejä.

Tutkimuksen empiirinen osa käsittää dataohjelman muodossa tehdyn kehittämistyön, jonka perustana on ohjelmointi Visual Basic-ohjelmassa. Tutkimuksen tavoitteena on luovalla tavalla yhdistää Microsoft Excel- ja Robot Structural Analysis- ohjelmien tarjoamia eri mahdollisuuksia.

Tulokset näyttävät, että on jokseenkin helppoa luoda rakentamisessa hyödyllisiä data-applikaatioita. Applikaatiot voivat huomattavasti helpottaa muuten aikaa vaativissa rutiinilaskelmien ja syöttöjen tekemisessä.

Kieli: ruotsi Avainsanat: kuormayhdistelyt, Microsoft Excel, Robot Structrual Analysis

---

BACHELOR'S THESIS

Author:	Jack Sandén
Degree Programme:	Construction Engineering, Vaasa
Specialization:	Structural Design
Supervisors:	Allan Andersson and Anders Borg

Title: *Import, modify and update – Microsoft Excel application for manual load case generation in Robot Structural Analysis*

Date Number of pages 32 Appendices 3

Summary

The purpose of this thesis is to develop and present a functional and useful computer application, which can efficiently and time-savingly generate and modify load combinations. Load combinations are simply translated into unfavorably combined loads on a structure. The theoretical background consists of the designing principles stated in Finland. A deeper knowledge of how to create these combinations is required to be able to identify possible enhancement options. The theoretical overview presents how load combinations are created using standardized instructions.

The empirical research consists of a computer application based on programming in a Visual Basic-environment. The research focuses on how to creatively combine the possibilities that Microsoft Excel and Robot Structural Analysis offer.

The results indicate that it is fairly simple to create computer applications that are useful in construction engineering and can considerably facilitate otherwise time consuming routine calculations and data inputs.

Language: Swedish Key words: load combinations, Microsoft Excel, Robot Structural Analysis

---

Innehållsförteckning

BILAGEFÖRTECKNING

1. Inledning

Idén till detta arbete uppstod av en ren tillfällighet. Hösten 2013 utförde jag företagsförlagd utbildning på konstruktionsbyrå Contria Ab i Vasa. Till ett av uppdragen hörde att, med hjälp av programmet Robot Structural Analysis Professional 2012, ta fram belastningar på en hallbyggnads grundkonstruktion. Uppgiften i sig var tidskrävande, men alltför stor del av tiden gick ändå åt till mekaniskt arbete såsom inmatning av data. I min frustration över detta började jag leta efter hjälpmedel och tips på programmets ”helpdesk” samt på forum anknutna till detsamma. Inte helt överraskande hittade jag flera andra som haft motsvarande problem och som även föreslagit effektiva lösningar på detta. Jag kunde sålunda notera att det fanns åtminstone två goda motiv för att granska frågan mera ingående; andra i branschen hade stött på likartade problem och försök till lösningsmodeller hade initierats. Att kunna generera och modifiera lastfallskombinationer på ett mera effektivt och tidsbesparande sätt borde för den skull vara ett relevant tema för ett forskningsarbete inom byggnadskonstruktion.

Föreliggande arbete är resultatet av mina ansträngningar att finna funktionella och användbara lösningar på detta. Övergripande har forskningsuppgiften således handlat om att effektivera byggnadsplaneringsprocessen och ytterst spara tid och pengar. För läsaren bör poängteras att detta arbete är en teoretisk beskrivning av funktionen hos det dataprogram som utarbetats inom ramen för forskningsarbetet.

Arbetet har jag strukturerat på följande sätt: Först ger jag en teoretisk bakgrund till hur lastfallkombinationerna tas fram med hjälp av standardiserade direktiv. Jag går därefter vidare till att kort beskriva de program jag använt mig av vid utarbetning av detta arbete för att sedan ge mig i kast med programmeringens språk och problematik. Avslutningsvis gör jag en sammanfattning med slutsatser och egna reflektioner.

Programmen som använts i detta arbete kommer framöver att förkortas på följande sätt:

Application Programming Interface	API
Microsoft Excel	Excel
Robot Structural Analysis Professional	Robot
Visual Basic	VB

2. Teoretisk bakgrund

För att verkningsfullt kunna utveckla och effektivera arbetsmomenten vid framtagning av lastfallskombinationer är det väsentligt att förstå hur och varför dessa kombinationer tas fram. I detta avsnitt redogörs för de teoretiska överväganden som ligger till grund för framtagning av lastfallskombinationer. Utgångspunkten utgörs mestadels av standardiserade direktiv.

2.1 Lastfallskombinationer enligt EN 1990 och nationella bilagan NA (FI)

Den svenska Eurokoden har använts i arbetet, men därutöver har Finlands nationella bilaga tillämpats. Enligt den svenska standarden SS-EN 1990 (2002), finns ett flertal olika typer av lastfallskombinationer. I mitt arbete har jag försökt möjliggöra användningen av samtliga lastfallskombinationer som gäller för Finland, vilket i detta fall utesluter bl.a. seismiska dimensioneringsprinciper. De vanliga kombinationerna uppgår därför till bruksgräns-, brottgräns-, olycksfalls- och brandkombinationer, vilka alla tillämpas i detta arbete.

De ovan nämnda kombinationerna är strukturerade enligt rubrikerna med motsvarande namn. All information om dessa uttagna ur EN 1990 (2002) samt RIL 201-1-2011 (2011). I och med detta har formler och uttryck numrerats med samma nummerkodning som i normerna för att förenkla kontroll och fördjupning.

2.1.1 Beteckningar

Beteckningarna är tagna ur EN 1990 (2002) kap. 1.6.

A_d	Olyckslast
E	Lasteffekt
E_d	Dimensionerande värde för lasteffekt
$G_{k,j}$	Karakteristiskt värde för den permanenta lasten j
P	Relevant representativt värde för spännkraft

P_k Karakteristiskt värde för spännkraft

$Q_{k,1}$ Karakteristiskt värde för en variabel huvudlast 1

γ_G Partialkoefficient för permanenta laster

γ_Q Partialkoefficient för variabellast

Ψ_0 Faktor för kombinationsvärde för variabel last

Ψ_1 Faktor för frekvent värde för variabel last

Ψ_2 Faktor för kvasipermanent värde för variabel last

2.1.2 Bruksgränstillstånd

Nedan följer en redogörelse av hur RIL 201-1-2011 (2011) beskriver vad bruksgränstillstånd innebär samt hur lastfallen kombineras i detta tillstånd.

Kapitel 6.5.1 i RIL 201-1-2011 (2011), som beaktar Finlands nationella bilaga, beskriver bruksgränstillstånd på följande sätt:

- Förskjutningar, som inverkar på utseendet, användarnas trivsel eller byggnadens användning.
- Vibrationer, som kan orsaka obehag hos människor eller förhindrar byggnadens användningsändamål.
- Skador, som tydligt verkar skadligt på utseende, beständighet och användning.

Bestående och återgående gränstillstånd bör dock åtskiljas.

Följande krav ställs på bruksgränstillstånd:

$E_d \leq C_d$, där (6.13)

E_d är dimensioneringsvärde för de lasteffekter som specificeras i brukbarhetskriteriet bestämt utgående från den aktuella lastkombinationen.

C_d är dimensionerande gränsvärdet för det aktuella brukbarhetskriteriet.

Brukbarhetskriterierna statueras i därpå följande kapitel 6.5.2 som i sin tur hänvisar till bilaga A ”Regler för byggnader”, i samma norm. Här tas brukbarhetskriterierna upp i kapitel A1.4.2 och lyder som följer:

(1) Bruksgränstillstånd bör beakta kriterier som berör, t.ex. styvhet hos bjälklag och yttertak, skillnader mellan golvnivåer samt vånings- och byggnadssvaj.
(2) Brukbarhetskriterier bör vara projektspecifika och överenskommas med byggherren.
(3) Brukbarhetskriterier för deformationer och svängningar skall definieras.

Dessa tre punkter kan ytterligare specificeras, men vidare redogörelse är ovidkommande i detta arbete.

Kapitel 6.5.3 i EN 1990 (2002) redogör för lastfallskombinationer i bruksgränstillstånd, vilket utgör detta arbetes väsentligaste del. Dessa beskrivs enligt följande:

(1) Lastfallskombinationer som beaktas i de aktuella dimensioneringssituationerna bör uppfylla de brukbarhetskriterier och funktionskriterier som statueras.
(2) Lastfallskombinationer för bruksgränstillstånd definieras symboliskt genom följande uttryck:

a) Karakteristisk kombination

$$E_d = E\{G_{k,j}; P; Q_{k,1}; \Psi_{0,i} Q_{k,i}\} \quad \text{där } j \geq 1; i > 1 \tag{6.14a}$$

som normalt tillämpas vid irreversibla gränstillstånd. Lastkombinationen inom parentesen kan uttryckas som:

$$\sum_{j \geq 1} G_{k,j} + P_k + Q_{k,1} + \sum_{i>1} \Psi_{0,i} Q_{k,i} \tag{6.14b}$$

b) Frekvent kombination

$$E_d = E\{G_{k,j}; P; \Psi_{1,i} Q_{k,1}; \Psi_{2,i} Q_{k,i}\} \quad \text{där } j \geq 1; i > 1 \tag{6.15a}$$

som normalt tillämpas vid reversibla gränstillstånd. Lastkombinationen inom parentesen kan uttryckas som:

$$\sum_{j\geq 1} G_{k,j} + P + \Psi_{1,1} Q_{k,1} + \sum_{i>1} \Psi_{2,i} Q_{k,i} \qquad (6.15b)$$

c) Kvasipermanent kombination

$$E_d = E\{G_{k,j}; P; \Psi_{2,i} Q_{k,i}\} \quad \text{där } j \geq 1; i > 1 \qquad (6.16a)$$

Lastkombinationen inom parentesen kan uttryckas som:

$$\sum_{j\geq 1} G_{k,j} + P_k + \sum_{i>1} \Psi_{2,i} Q_{k,i} \qquad (6.16b)$$

Som vanligen tillämpas vid långtidseffekter och effekter rörande bärverkets utseende.

Partialkoefficienterna γ_M för material bör enligt EN 1990 (2002) kap. 6.5.4 sättas till 1,0 så länge inget annat anges i EN 1992 t.o.m. 1999. Kombinationsfaktorn Ψ fås från tabellen nedan.

På sidorna 11–13 redogörs ytterligare för innebörder av olika lastfallskombinationer. De största skillnaderna mellan kombinationerna utgörs av varierande säkerhetsfaktorer beroende på typ av byggnadskonstruktion.

Tabell 1. Kombinationsfaktor Ψ för byggnader enligt den finska nationella bilagan NA(FI). Källa: RIL-201-1-2011(2011) Tabell A1.1(FI)

Tabell A1.1 (FI): Värden på koefficienter ψ för byggnader

Last	ψ_0	ψ_1	ψ_2
Nyttiga laster i byggnader, klass (se SFS-EN 1991-1-1)			
Klass A: bostadsutrymmen	0,7	0,5	0,3
Klass B: kontorsutrymmen	0,7	0,5	0,3
Klass C: samlingsutrymmen	0,7	0,7	0,3
Klass D: affärsutrymmen	0,7	0,7	0,6
Klass E: lagerutrymmen	1,0	0,9	0,8
Klass F: trafikerade utrymmen, fordonsvikt ≤ 30 kN	0,7	0,7	0,6
Klass G: trafikerade utrymmen, 30 kN < fordonsvikt ≤ 160 kN	0,7	0,5	0,3
Klass H: yttertak	0	0	0
Snölast (se SFS-EN 1991-1-3)*) när			
s_k < 2,75 kN/m^2	0,7	0,4	0,2
s_k ≥ 2,75 kN/m^2	0,7	0,5	0,2
Islast **)	0,7	0,3	0
Vindlaster på byggnader (se SFS-EN 1991-1-4)	0,6	0,2	0
Byggnaders inre temperatur (ej brand) (se SFS-EN 1991-1-5)	0,6	0,5	0

*) På uteterrasser och balkonger ψ_0 = 0 i samband med klasserna A, B, F och G.
Obs: Om det i byggnaden finns olika lastklasser som inte kan separeras till egna klara grupper, används ψ-värden som ger mest ogynnsam inverkan.
**) Tillägg till Finlands nationella bilaga.

Kombinationsfaktorerna Ψ används enligt följande (EN 1990 (2002), s.15):

- Kombinationsvärde för en variabel last ($\Psi_0 Q_k$) väljs så att sannolikheten att de effekter som orsakas av kombinationen kommer att överskridas är ungefär densamma som för karakteristiska värdet för en individuell last.
- Det frekventa värdet för en variabel last ($\Psi_1 Q_k$) bestäms så att antingen den totala tiden inom referensperioden under vilket värdet överskrids utgörs endast en liten del av referensperioden, eller så att frekvensen av ett överskridande begränsas till ett angivet värde.
- Det kvasipermanenta värdet ($\Psi_2 Q_k$) bestäms så att antingen den totala tiden inom referensperioden under vilket värdet överskrids utgörs av endast en liten angiven del av referensperioden, eller så att frekvensen av ett överskridande begränsas till ett angivet värde.

2.1.3 Brottgränstillstånd

I enlighet med Eurokod EN 1990 (2002) kapitel 6.4 skall aktuellt brottgränstillstånd verifieras utgående från:

a) EQU: förlorad statisk jämvikt för bärverket när det betraktas som en stel kropp;
b) STR: Inre brott eller för stor deformation av bärverket eller bärverksdelarna;
c) GEO: Brott eller för stor deformation i undergrunden med inverkan på bärverkets bärförmåga;
d) FAT: Brott p.g.a. utmattning hos bärverket eller bärverksdelarna.

EN 1990 (2002) kapitel 6.4.1

Lastfallkombinationerna under kapitelnummer 6.4.3 delas in i underrubriker:

I punkt 6.4.3.1 i normen behandlas lastfallskombinationer i brottgränstillstånd. Följande regler gäller:

(1) För varje kritiskt lastfall skall dimensioneringsvärden för lasteffekter (E_d) bestämmas genom att kombinera lastvärden som verkar samtidigt.
(2) Varje lastkombination bör bestå av:
- en variabel last som huvudlast eller
- en olyckslast.
(3) Lastkombinationerna bör väljas enligt kap. 6.4.3.2 och 6.4.3.3 enligt normerna, vilket innefattas av de följande två kapitlen i detta arbete.
(4) Där verifieringsresultaten är variationskänsliga bör gynnsamma och ogynnsamma laster räknas individuellt.
(5) Där flera effekter av en last inte är fullt korrelerade, kan partialkoefficienten som tillämpas på en gynnsam komponent reduceras.
(6) Påtvingade deformationer bör beaktas i förekommande fall.

Varaktiga eller tillfälliga dimensioneringssituationer ser enligt kapitel 6.4.3.2 ut på följande sätt:

(1) Det allmänna uttrycket för lasteffekter som bör tillämpas är:

$$E_d = \gamma_{Sd} E\{\gamma_{g,j} G_{k,j}; \Upsilon_p P; \gamma_{q,1} Q_{k,1}; \gamma_{q,i} \Psi_{0,i} Q_{k,i}\} \quad d\bar{a}r\ j \geq 1; i > 1 \qquad (6.9a)$$

(2) De kombinationer av lasteffekter som beaktas baseras på

- dimensioneringsvärdet för den variabla huvudlasten och
- dimensionerande kombinationsvärden för samverkande variabla laster

$$E_d = E\{\gamma_{g,j} G_{k,j}; \Upsilon_p P; \gamma_{Q,1} Q_{k,1}; \gamma_{Q,i} \Psi_{0,i} Q_{k,i}\} \quad d\bar{a}r\ j \geq 1; i > 1 \qquad (6.9b)$$

(3) Lastkombinationen inom parentesen kan antingen uttryckas som

$$\sum_{j\geq 1} \gamma_{Gj} G_{k,j} + \gamma_P P_k + \gamma_{Q,1} Q_{k,1} + \sum_{i>1} \gamma_{Q,i} \Psi_{0,i} Q_{k,i} \qquad (6.10)$$

eller, som alternativ för gränstillstånden STR och GEO, det minst gynnsamma av uttrycken:

$$\sum_{j\geq 1} \gamma_{Gj} G_{kj} + \gamma_P P_k + \gamma_{Q,1} \Psi_{0,i} Q_{k,1} + \sum_{i>1} \gamma_{Q,i} \Psi_{0,i} Q_{k,i} \qquad (6.10a)$$

och

$$\sum_{j\geq 1} \xi_j \gamma_{Gj} G_{k,j} + \gamma_P P_k + \gamma_{Q,1} Q_{k,1} + \sum_{i>1} \gamma_{Q,i} \Psi_{0,i} Q_{k,i} \qquad (6.10b)$$

Där:

”+” betyder ”att kombineras med”

Σ betyder ”den kombinerade effekten av”

ξ är en reduktionsfaktor för ogynnsamma permanenta laster G

(4) Om sambandet mellan laster och dess effekter inte är linjärt, bör de båda uttrycken ovan tillämpas direkt beroende på den relativa ökningen av lasternas storlek.

Kapitel 6.4.3.3 går in på exceptionella dimensioneringssituationer såsom påkörning, brand eller fortsatt beständighet efter dylik händelse.

(1) Det allmänna uttrycket för lasteffekter bör vara:

$$E_d = E\{G_{k,j}; P; A_d; (\Psi_{1,1} \text{ eller } \Psi_{2,1})Q_{k,1}; \Psi_{2,i}Q_{k,i}\} \quad \text{där } j \geq 1; i > 1 \tag{6.11a}$$

(2) Lastkombinationen inom parentesen kan uttryckas som:

$$\sum_{j\geq 1} G_{k,j} + P + A_d + (\Psi_{1,1} \text{ eller } \Psi_{2,1})Q_{k1} + \sum_{i>1} \Psi_{2,i}Q_{k,i} \tag{6.11b}$$

(3) Valet mellan $\Psi_{1,1}Q_{k,1}$ eller $\Psi_{2,1}Q_{k,1}$ bör relateras till den aktuella exceptionella dimensioneringssituationen.

(4) Lastkombinationer för exceptionella dimensioneringssituationer bör:

- innefatta en uttrycklig olyckslast A (brand eller påkörning), eller
- hänföra sig till en situation efter en olyckshändelse (A = 0).

Partialkoefficienterna (γ och Ψ) som använts i kombinationerna för laster hänvisas i EN 1990 till EN 1991 och bilaga A, där Ψ illustreras i *Tabell 1* och γ i *Tabell 2–4* i detta arbete. Material och produkter hänvisas till EN 1992 t.o.m. EN 1999.

Tabell 2. Beroende på byggnadens konsekvensklass multipliceras lastfallskombinationen med en av nedanstående faktorer. Konsekvensklassernas definition i nationella bilagans B-del. Källa: Tabell A1.2(B) i nationella bilagan.

K_{FI} beror på tillförlitlighetsklassen enligt tabell B2 i bilaga B i standarden SFS-EN 1990 på följande sätt: i tillförlitlighetsklass RC3 $K_{FI} = 1{,}1$ i tillförlitlighetsklass RC2 $K_{FI} = 1{,}0$ i tillförlitlighetsklass RC1 $K_{FI} = 0{,}9$. Konsekvensklasser CC3-CC1 som klargör tillförlitlighetsklasserna presenteras i bilaga B.

Tabell 3. Dimensioneringsvärden för ekvation 6.10 (EQU) enligt kapitel A1.3.1 i den nationella bilagan.

Tabell A1.2(A) (FI) Dimensioneringsvärden för laster (EQU) (Serie A)

Vanliga och tillfälliga dimensioneringssituationer	Permanenta laster		Dimensionerande variabel last (*)	Andra samtidiga variabla laster (*)
	Ogynnsamma	Gynnsamma		
(Ekv. 6.10)	$1{,}1\ K_{FI} G_{kj,sup}$	$0{,}9\ G_{kj,inf}$	$1{,}5\ K_{FI}\ Q_{k,1}$	$1{,}5\ K_{FI}\ \psi_{0,i} Q_{k,i}$

Tabell 4. Med avseende på ekvationerna 6.10, 6.10a samt 6.10b i normerna EN 1990-1-1 + NA(FI) fås dimensioneringsvärden ur tabellen.

Tabell A1.2(B) (FI) Dimensioneringsvärden för laster (STR/GEO) (Serie B)

Vanliga och tillfälliga dimensioneringssituationer	Permanenta laster		Dimensionerande variabel last (*)	Andra samtidiga variabla laster (*)
	Ogynnsamma	Gynnsamma		
(Ekv. 6.10a)	$1{,}35\ K_{FI}\ G_{kj,sup}$	$0{,}9\ G_{kj,inf}$		
(Ekv. 6.10b)	$1{,}15\ K_{FI}\ G_{kj,sup}$	$0{,}9\ G_{kj,inf}$	$1{,}5\ K_{FI}\ Q_{k,1}$	$1{,}5\ K_{FI}\ \psi_{0,i} Q_{k,i}$

(*) Laster enligt tabell A.1.1 är variabla laster.

Anm. 1: Som dimensioneringsformel kan saken uttryckas så att som lastkombination används det mera ogynnsamma av de två följande uttrycken, varvid det skall observeras att det senare uttrycket endast innehåller permanenta laster:

$$\begin{cases} 1{,}15\,K_{FI}\,G_{kj,sup} + 0{,}9\,G_{kj,inf} + 1{,}5\,K_{FI}\,Q_{k,1} + 1{,}5\,K_{FI}\sum_{i>1}\psi_{0,i}\,Q_{k,i} \\ 1{,}35\,K_{FI}\,G_{kj,sup} + 0{,}9\,G_{kj,inf} \end{cases}$$

K_{FI} beror på tillförlitlighetsklassen enligt tabell B2 i bilaga B i standarden SFS-EN 1990 på följande sätt:

i tillförlitlighetsklass RC3 $K_{FI} = 1{,}1$
i tillförlitlighetsklass RC2 $K_{FI} = 1{,}0$
i tillförlitlighetsklass RC1 $K_{FI} = 0{,}9$.

Konsekvensklasser CC3-CC1 som klargör tillförlitlighetsklasserna presenteras i bilaga B.

Anm. 2: Se även i standarderna SFS-EN 1992 - SFS-EN 1999 de γ -värden för partialsäkerhetskoefficient som används vid tvångsövergångs- eller tvångsdeformationstillstånd

Anm. 3: De karakteristiska värdena för alla permanenta laster som har samma grund multipliceras med partialsäkerhetstalet $\gamma_{G,sup}$, om lastens sammanlagda inverkan är ogynnsam och med partialsäkerhestalet $\gamma_{G,inf}$, om lastens sammanlagda inverkan är gynnsam. Till exempel kan alla laster orsakade av konstruktionens egen vikt anses ha samma grund; detta gäller även om det är frågan om olika material.

Anm. 4: Vid specialgranskningar kan värden för partialsäkerhetstalen γ_G och γ_Q delas upp i γg och γq samt modellens osäkerhetskoefficient γSd. I de flesta fall kan ett värde mellan 1,05 och 1,15 användas för osäkerhetskoefficienten γSd.

Anm. 5: För geoteknisk dimensionering av grundkonstruktioner se standard SFS-EN 1997-1 med sina nationella bilagor.

Tabell 5. Dimensioneringsvärden för ekvation 6.10 (STR/GEO) enligt kapitel 6.4.3.2 i normen EN 1990 (2002).

Tabell A1.2(C) (FI) Dimensioneringsvärden för laster (STR/GEO) (Serie C)

Vanliga och tillfälliga dimensionerings-situationer	Permanenta laster		Dimensionerande variabel last (*)	Andra samtidiga variabla laster (*)
	Ogynnsamma	Gynnsamma		
(Ekv. 6.10)	1,0 K_{FI} $G_{kj,sup}$	1,0 $G_{kj,inf}$	1,3 K_{FI} $Q_{k,1}$	1,3 K_{FI} $\psi_{0,i}Q_{k,i}$

(*) Laster enligt tabell A.1.1 är variabla laster

K_{FI} beror av tillförlitlighetsklassen enligt tabell B2 i bilaga B på följande sätt:

i tillförlitlighetsklass RC3 $K_{FI} = 1,1$
i tillförlitlighetsklass RC2 $K_{FI} = 1,0$
i tillförlitlighetsklass RC1 $K_{FI} = 0,9$.

Konsekvensklasser CC3-CC1 som klargör tillförlitlighetsklasserna presenteras i bilaga B.

Utgående från ovanstående tabeller och uttryck kombinerar man ett aktuellt fall enligt värsta tänkbara kombination. Nedan följer tillämpningen av dessa formler enligt RIL 201-1-1-2011 (2011), som numrerar ekvationerna enligt EN 1990 (2002), med undantag av bokstaven ”S” för indikering av Finlands nationella bilagas kombinationer.

I enlighet med tabell 3 med avseende på statisk jämvikt enligt uppsättning A (EQU) kan formeln utryckas som:

$$\begin{matrix}1,1\ K_{FI} \\ 0,9\end{matrix}\Big\}\sum_{j\geq 1} G_{k,j} + \gamma_P P + 1,5 K_{FI} Q_{k,1} + 1,5 K_{FI} \sum_{i>1} \Psi_{0,i} \cdot Q_{k,i} \qquad (6.9S)$$

med beaktande av konsekvensklass och lastfaktorer så att:

- de ogynnsamma bestående lasterna som försvagar jämvikten (G_k) multipliceras med faktorn $1,1K_{FI}$
- de gynnsamma bestående lasterna som förbättrar jämvikten (G_k) multipliceras med 0,9 (observera utan K_{FI})
- de förspända krafterna P multipliceras med partialkoefficienten γ_P, som anges i EN 1992 samt i kapitel 2.4.1.1 samt 2.4.2.2 i den nationella bilagan.

- den dimensionerande nyttolasten $Q_{k,1}$ multipliceras med faktorn 1,5 K_{FI}
- övriga samtidigt-verkande nyttolaster $Q_{k,i}$ multipliceras med sin kombinationsfaktor ($\Psi_0 Q_i$) samt faktorn 1,5 $K_{FI.}$ Förenklat kan sägas att kombinationsfaktorn reducerar gynnsamt verkande belastningar.

På motsvarande vis, dock enligt uppsättning B i tabell 4, ställs uttrycket för byggnadsdelarnas hållbarhet samt geotekniska bärighet (STR) upp på följande sätt:

$$\begin{matrix}1{,}15\ K_{FI}\\ \mathbf{0{,}9}\end{matrix}\Big\}\sum_{j\geq 1} G_{k,j} + \gamma_P P + 1{,}5 K_{FI} Q_{k,1} + 1{,}5 K_{FI} \sum_{i>1} \Psi_{0,i} \cdot Q_{k,i}$$

dock minst (6.10S)

$$\begin{matrix}1{,}35\ K_{FI}\\ \mathbf{0{,}9}\end{matrix}\Big\}\sum_{j\geq 1} G_{k,j}$$

med beaktande av konsekvensklass och lastfaktorer så att:

- de ogynnsamma bestående lasterna som försvagar jämvikten (G_k) multipliceras med faktorn 1,15 K_{FI}
- de gynnsamma bestående lasterna (G_k) multipliceras med 0,9 (observera utan K_{FI})
- de förspända krafterna P multipliceras med partialkoefficienten γ_P
- den dimensionerande nyttolasten $Q_{k,1}$ multipliceras med faktorn 1,5 K_{FI}
- övriga samtidigt-verkande nyttolaster $Q_{k,i}$ multipliceras med sin kombinationsfaktor ($\Psi_0 Q_i$) samt faktorn 1,5 K_{FI}
- formeln används även vid beräkning av byggnadsdelarnas hållfasthet, när grundens bärighet och geotekniska belastningar inverkar.

Den geotekniska bärigheten (GEO) uttrycks med hjälp av uppsättning C i tabell 5 enligt följande:

$$\begin{matrix}1{,}0\ K_{FI}\\ \mathbf{1{,}0}\end{matrix}\Big\}\sum_{j\geq 1} G_{k,j} + \gamma_P P + 1{,}3 K_{FI} Q_{k,1} + 1{,}3 K_{FI} \sum_{i>1} \Psi_{0,i} \cdot Q_{k,i} \qquad (6.11S)$$

med beaktande av konsekvensklasser och lastfaktorer så att:

- de ogynnsamma bestående lasterna som försvagar bärförmågan (G_k) multipliceras med faktorn 1,0 K_{FI}
- de gynnsamma bestående lasterna (G_k) multipliceras med 1,0 (observera utan K_{FI})

- de förspända krafterna P multipliceras med partialkoefficienten γ_P
- den dimensionerande nyttolasten $Q_{k,1}$ multipliceras med faktorn 1,3 K_{FI}
- övriga samtidigt-verkande nyttolaster $Q_{k,i}$ multipliceras med sin kombinationsfaktor ($\Psi_0 Q_i$) samt faktorn 1,3 K_{FI}
- glidning och stabilitet tas med i beräkningarna.

Till skillnad från de ovanstående lastuttrycken beskrivs olycksfall genom följande upplägg:

När snö-, is- och vindlast räknas till huvudlast (Q_{k1}).

$$\sum_{j\geq 1} G_{kj} + P + A_d + \Psi_{11} Q_{k1} + \sum_{i>1} \Psi_{2,j} Q_{k,i} \qquad (6.12S)$$

Om huvudlasten (Q_{k1}) är något annat än snö-, is- och vindlast.

$$\sum_{j\geq 1} G_{kj} + P + A_d + \Psi_{21} Q_{k1} + \sum_{i>1} \Psi_{2,j} Q_{k,i} \qquad (6.13S)$$

I uttrycket räknas följande samman:

- de ogynnsamma bestående lasterna
- de gynnsamma bestående lasterna
- de förspända krafterna P
- den dimensionerande nyttolasten med kombinationsfaktor ($\Psi_{11} Q_{k,1}$ eller $\Psi_{21} Q_{k,1}$)
- övriga samtidigt-verkande nyttolaster multipliceras med sin kombinationsfaktor ($\Psi_{2,i} Q_{k,i}$).

2.2 Datastöd och byggnadskonstruktion

I och med detta arbete har nya begrepp och programvaror aktualiserats. Nedan följer några exempel på dessa program och metoder samt en kort redogörelse av deras innehåll.

2.2.1 FEM (Finite Element Method)

Orsaken till att den finita elementmetoden behandlas är att Robot bygger på denna metod. Därför bör det anses viktigt att reda ut några enkla begrepp, utan att desto närmare gå in på ämnet. Den finita elementmetoden har behandlats ingående i samband med utbildningens kurser inom hållfasthetslära och mekanik. Bengt Nilsson (2013) gör, i kompendiet ”Finita elementmetoden – En kort introduktion till teorin”, en generell genomgång av viktiga och grundläggande begrepp när man talar om finita elementmetoden.

Enligt Nilsson är finita elementmetoden, eller FEM som den ofta förkortas, en generell matematisk och numerisk metod för att söka approximativa lösningar för komplicerade för att lösas med vanliga metoder. Det finns olika modeller för lösning av strukturer med finita element. Lösningen går ut på att strukturen indelas i små element med kända egenskaper och grundar sig på materialens styvhetsegenskaper eller på utfört arbete. Metoden används ofta vid s.k. fältproblem som ofta förknippas med hållfasthetsberäkningar.

Karakteristiskt för metoden är att geometrin delas upp, eller diskretiseras, i små stycken med enkel geometri s.k. finita element, skriver Nilsson. Typiskt för FEM är att elementen är bundna med varandra via speciella punkter. Dessa punkter kan kallas noder, vilket för övrigt används i Robot. Elementen består oftast av trianglar eller fyrhörningar vid tvådimensionella beräkningar. Kombinationen av element och noder kallas nät eller ”mesh” på engelska, som även används i Robot.

2.2.2 Robot Structural Analysis Professional

Robot är en programvara avsett för konstruktörer för avancerade simuleringar och analysmöjligheter för byggnader och strukturer. Programmet följer valbart landspecifika normer och erbjuder modelleringsmöjligheter i betong, stål och trä. Som tidigare nämnt använder programmet sig av FEM-metoden, vilket möjliggör avancerade beräkningar och analyser. Användningen begränsas dock ofta till hållfasthetsberäkningar eftersom det visuella och estetiska perspektivet begränsas av de matematiska funktionerna. Generering av ritningar går heller inte att utföra i Robot, men möjliggörs genom exportering till andra program såsom Tekla Structures. Programmet erbjuder även en automatisk generering av

lastfallskombinationer normenligt. Detta är effektivt vid planering med få lastfall, men gör modellen långsam när lastfallen är många.

Robot tillhör BIM-familjen, vilket står för ”Building Information Modeling” som blivit ett vanligt begrepp inom byggnadsplaneringen.

2.2.3 Visual Basic

I en artikel på hemsidan ”*About.com*” gör Dan Mabbutt en kort redogörelse om vad Visual Basic handlar om. VB skapades, enligt Mabbutt, av Microsoft 1991. Tanken med programmet var att på ett betydligt enklare och snabbare sätt kunna skapa program till det nya grafiska Windows-operativsystemet. Tidigare hade program i allmänhet endast programmerats med språket C++, vilket var dyrare och svårare att använda samt innehöll en stor mängd buggar.

Efter noga jämförelse med andra källor valde jag att även plocka ut fragment om VB från Wikipedia, eftersom denna sida överskådligt summerar programmets historik.

Enligt Wikipedia bygger VB på ett språk som från början kallades Beginner’s All-purpose Symbolic Instruction Code, eller BASIC utvecklat år 1964. Programmet framtogs för att universitetsstuderanden lättare skulle få förståelse för grundläggande programmeringsprinciper. Visual Basic togs i bruk för att öppna porten till de visuella operativsystemen och har därefter utvecklats fram till VB 6.0. (Visual Basic (u.å.))

3. Syfte och problemprecisering

Inom ramen för en praktikperiod hösten 2013 ingick som ett uppdrag att framta belastningar som uppstår i en halls grundkonstruktion. Till hjälp för dylika problem användes programmet Robot Structural Analysis Professional 2012. En tidig iakttagelse var att största delen av tiden gick åt till inmatning av befintlig data i programmet och inte till modellering av själva hallen. Grunden till detta låg till stor del i att specifika lastfall och lastfallskombinationer skulle kontrolleras. En sökning på nätet visade även att andra i

konstruktionsbranschen identifierat motsvarande utmaningar. Utifrån problemformuleringen uttrycktes som övergripande syfte att finna mer tids- och kostnadseffektiva metoder vid framtagning av lastfallskombinationer. Därmed kunde tid och pengar sparas.

Utifrån syftet har följande tre forskningsfrågor formulerats:

1. På vilket sätt kunde rutinmässiga uppgifter vid lastfallsberäkningar ersättas med hjälp av dataprogram?
2. Vilka kvalitativa förbättringar kunde ett dylikt dataprogram bidra med?
3. Hur kunde ett dylikt datorprogram utformas?

Mer specifikt består forskningen således i att söka och skapa kreativa lösningar på hur tidsdrygt och mekaniskt rutinarbete kan underlättas och effektiveras.

Robot erbjuder möjlighet till automatisk generering av lastfallskombinationer, men skapar därmed även stora mängder med kombinationer som genast kan förkastas. Detta i sin tur kan leda till ett flertal onödiga kombinationer beroende på antalet lastfall man utgår ifrån. En alltför stor mängd kombinationer gör programmet långsamt vid kalkylering på grund av alla lastfallskombinationer som skall räknas igenom. Därför skulle önskade lastfallskombinationer inmatas manuellt, vilket fort visade sig både arbets- och tidsdrygt när lastfallen uppgick till elva. Ett flertal timmar gick därmed till denna inmatning.

När jag senare, av helt andra orsaker, råkade på en guide gällande användningen av API (Application Programming Interface) i Robot förföll det sig naturligt att tillämpa detta på mitt tidigare dilemma.

Det mest påtagliga bekymret jag stött på under arbetets gång är mina begränsade programmeringskunskaper. Detta har lett till avgränsningar i funktion hos programmet, samt en stor tidsåtgång vad gäller informations- och problemsökning. Tack vare en välfungerande helpdesk och ett ihärdigt googlande har ändå ett fungerande program uppnåtts, om än med otaliga brister. Med hjälp av ett flertal guider och programmallar är det möjligt att utvidga programmet med flera olika funktioner. På grund av begränsad tid riktades dock fokus till lastfall och lastfallskombinationer, som kan anses vara mest tidsödande vid modellering i Robot.

4. Metoder

En förutsättning för detta arbetes genomförande har varit insikter i byggnadskonstruktion och mer specifikt lastfallskombinationer. För att uppnå forskningens syfte har dock insikter i programmering, som utfördes med hjälp av Visual Basic, varit nödvändiga och som beskrivits i korthet i kapitel 2.2.3. Mina tidigare kunskaper av programmet var som tidigare noterats näst intill obefintliga och de programmeringskunskaper jag hade i bagaget bestod uteslutande av Python-programmering. Grundprinciperna visade sig dock vara likartade och tack vare ett intensivt googlande och rådfrågande på diverse programmeringsforum erhölls en riktgivande uppfattning av hur uppbyggnaden av dylikt program skulle utföras.

Som primärkälla användes en guide enkom framtagen för informationsbyte mellan Microsoft Excel och Robot Structural Analysis, vilken laddades ner från forums.autodesk.com. Guiden instruerar hur man, med hjälp av t.ex. Visual Basic, skapar sitt första program.

	Declare and create the main object representing the Robot application.
	`Dim robapp As IRobotApplication` `Set robapp = New RobotApplication`
	Start the calculation.
	`robapp.Project.CalcEngine.Calculate`
	Free all declared variables.
	`Set robapp = Nothing`

Figur 1. Programmeringsexempel på hur man skapar ett program i Visual Basic. Källa: API Tutorial (2010)

En genomgång av guiden blev startskottet till programmet som sedan skapades. Under vägens gång hittades dock programmeringsmallar och program med motsvarande funktion som eftersträvades. Programmet kom därför att bli ett hopplock av ett antal program, med fokus för mitt forskningsintresse: lastfallskombinationer. Valet av lastkombinationer innebar i sin tur att jag, logiskt sett, tvingades integrera lastfallen i programmet för att förmå programmen samverka. De program jag använt som mall är skrivna av Rafal Gaweda, API-specialist och produktstödperson på Autodesk, och hittas på samma adress som guiden som nämns ovan.

4.1 Programmering och kodning

I detta avsnitt redogörs inledningsvis för några ofta, i min kod, återkommande kommandon i VB. Detta för att förkorta, men framför allt förtydliga och åskådliggöra några av de viktigaste och enklaste funktionerna vid kodning. Detta åtföljs av ett antal viktiga kommandon och funktioner avsedda för aktivering och importering av Robot-data.

4.1.1 Visual Basic-kommandon

Med hjälp av kombinationer av dessa kommandon har programmet byggts i VB.

As Integer	Definierar heltal.
As Long	Används vid förvaring av stora siffervärden (9,22e^10).
As String	Används för förvaring av upp till 2^31 tecken.
Dim	Tilldelar och allokerar förvaringsutrymme för en eller fler variabler (lastfall, lasttyper, koefficienter osv.).
Do…Loop…Exit Do	Upprepar ett stycke/block medan resultatet är sant (True) eller tills resultatet blir sant (True).
For x To y	Starten på en for-slinga som upprepas tills värdet y nås. Slingans slut definieras med Next.
If…Then…Else	If ställer krav på ett påstående som, om uppfylls, efterföljs av Then med aktionskommando. Ifall påståendet inte uppfylls förflyttas man till Else med aktionskommando.
Set	Skapar ett objekt.
Sub	Anger namn på subrutin.

Källa: Microsoft Developer Network (18.02.2014)

4.1.2 Robot Structural Analysis Professional API-kommandon

Nedan följer ett antal VB-kommandon tagna från Autodesks API-tutorial för aktivering av särskilda funktioner. Samtliga nedanstående kommandon hittas även i min egen kodning. Kommandona består av en kombination av VB-kommandon samt Robots specificerade aktiveringsbegrepp. Koden illustreras med vanlig text, medan förklaringen placeras ovan koden med kursiverad stil.

Följande koder är tagna från Robots API-guide:

Skapa ett nytt robotobjekt
Dim robapp As IRobotApplication
Set robapp = New RobotApplication

Rensa minnet efter att programmet avslutats
Set robapp = Nothing

Skapa ett nytt lastfall
Dim SC As IRobotSimpleCase
Set SC = robapp.Project.Structure.Cases.CreateSimple(nummer, namn, lasttyp, linjär/olinjär)

Importera alla lastfall
Dim RCC As RobotCaseCollection
Set RCC = RobApp.Project.Structure.Cases.GetAll

Importera kombinationer
Dim Rcomb As IRobotCaseCombination

Kontrollera att Robot är igång
If Not RobApp.Visible
Then Robapp = Nothing

Dessa koder har tagits från Autodesks forum:

Skapar och förvarar kombinationstyper (ULS, SLS, ACC och FIRE)

```
Dim CT As IRobotCombinationType
```

Importera kombinationernas säkerhetskoefficienter

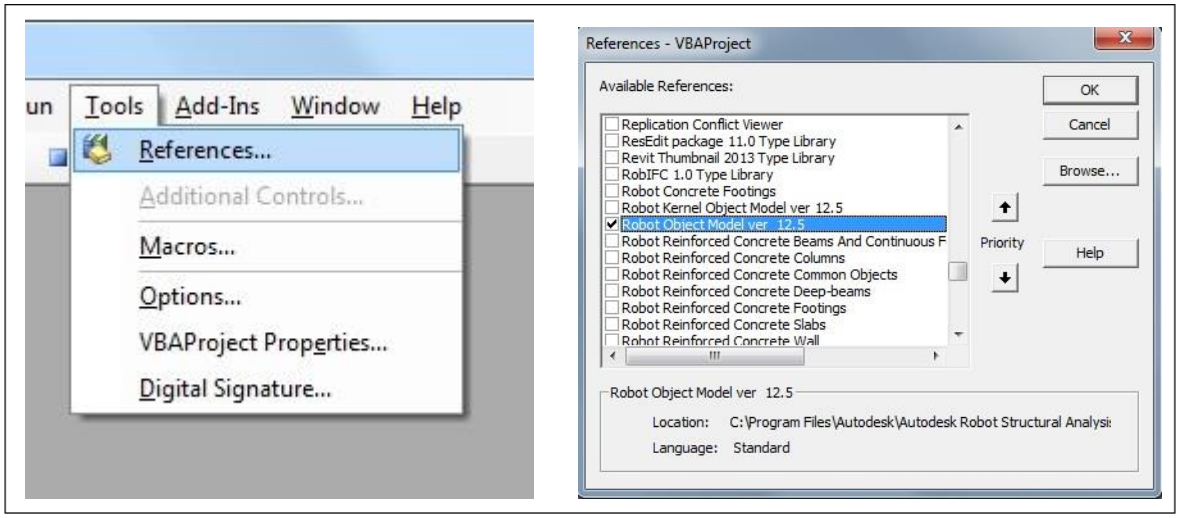
```
Dim RCCGP As RobotCodeCmbGenerationParams
Set RCCGP = RobApp.Project.Structure.Cases.CodeCmbEngine.Params
```

4.1.3 Kod och kodtolkning

Importen av lastfall från Robot till Excel visade sig vara betydligt enklare än importen av lastfallskombinationer. Nedan redogörs i korthet för väsentliga funktioner och kommandon som använts vid aktiveringen och importeringen av lastfallsdata i programmet. Därefter redogörs för detaljer kring importeringen av lastfallskombinationer och dess säkerhetsfaktorer. Det bör understrykas för läsaren att koden är skriven av en lekman och därmed kan innehålla fel och överflödig kod.

Koden presenteras i form av skärmdumpar från VB, istället för inskrivning av kod i löpande text. Därmed blir det lättare för läsaren att kunna plocka ut väsentliga avsnitt.

Innan man börjar skriva kod i detta syfte är det viktigt att kontrollera att referensdata för Robot är aktiverat i VB. Utan denna aktivering fungerar inte förflyttning mellan programmen. Aktiveringen sker via verktygs- och referensfunktionen i VB. Referensfilen ”Robot Object Model ver 12.0” letas upp i listan, markeras och aktiveras. Filnamnet kan dock variera beroende på vilken version av Robot som används.



Figur 2. Skärmdump ur Basic. Menyval (till vänster) och aktivering av referensfil.

4.1.3.1 Lastfall

Inledande VB-koden (*Figur 3)* för importering av lastfall återfinns i början av samtliga importeringskodningar, dvs. lastfall, lastfallskombinationer och linjelaster.

```
Private Sub Importera_lastfall_Click()

If MsgBox("Är du säker på att du vill importera?", vbYesNo) = vbNo Then Exit Sub

Dim RobApp As IRobotApplication
Set RobApp = New RobotApplication

If Not RobApp.Visible Then
   Set RobApp = Nothing
   MsgBox "Var vänlig och öppna en modell i Robot först!", vbOKOnly
   Exit Sub
End If
```

Figur 3. Urklipp ur VB-kodning för import av lastfall. Fullständig kod i bilaga 1.

De första två orden ”Private” och ”Sub” innebär aktivering av en procedur, med vars hjälp man skapar funktioner som t.ex. kan importera data. Knappen ”Importera lastfall från Robot” i ME har givits namnet ”Importera_lastfall” i VB och aktiverar proceduren vid knapptryckning.

Andra raden i *Figur 3* aktiverar en textruta som frågar om användaren är säker på att han/hon vill importera. Detta förhindrar att man oavsiktligt påbörjar ny, och därmed raderar gammal, import.

De följande två kommandona innebär att ett nytt objekt skapas och är tagna direkt ur API-guiden (*Figur 1*). Samtliga program för programmering mellan VB och Robot kräver dessa två rader.

De sista fem raderna kontrollerar att en Robot-modell är öppen, vilket krävs för att programmet skall fungera. Om så inte är fallet aktiveras en textruta som ombeder användaren att öppna en Robot-modell.

```
Dim RCC As RobotCaseCollection
Set RCC = RobApp.Project.Structure.Cases.GetAll
idx = 3

Range("A3", "E100").ClearContents

Dim Names(6) As String

[E1].Value = "Tillgängliga belastningar"

Set RCase = RobApp.Project.Structure.Cases.CreateSimple(100000, "tmp", 0, I_CAT_STATIC_LINEAR)
For i = 0 To 6
   RCase.Nature = i
   Names(i) = RCase.NatureName
Next i

RobApp.Project.Structure.Cases.Delete (100000)

Dim RCCGP As RobotCodeCmbGenerationParams
Set RCCGP = RobApp.Project.Structure.Cases.CodeCmbEngine.Params
```

Figur 4. Introduktion till importering av lastfall samt skapande av s.k. ”simple case”

De inledande tre raderna i *Figur 4* aktiverar funktionen för lastfallsdatabasen i Robot och preciserar importplatsens start till ruta rad nummer tre. För att enklast möjligt uppdatera gammal information valde jag att lägga in kommandot för radering av samtliga data före ny importering. Raderingen begränsas dock mellan cell A3 och E100.

Med hjälp av Autodesks forum (forums.autodesk.com) hittade jag VB-kod, skriven av Rafal Gaweda, med vars hjälp jag även kunde importera s.k. ”subnatures” eller lasttyper (vind-, snö-, nyttolast etc.). Med kommandot ”*for i = 0 to 6*” söker koden igenom

lasttyperna med sex upprepningar, eftersom det endast finns sex olika lasttyper i Robot med finska normer. Allteftersom lasttyperna gås igenom skrivs de ut som en lista under rubriken ”Tillgängliga belastningar” för att underlätta specificering av lasttyp när man vid senare tillfälle eventuellt skapar nya lastfall. Denna utskrift sker dock med hjälp av de första fyra raderna i koden nedan.

Värt att notera vid importering är att olika Robot-versioner kan ha olika namn för t.ex. sina strukturella belastningar. En version kan benämna dessa som ”STRC” medan en annan använder ”dead”. Vid senare modifiering är det viktigt att kontrollera att man namngivit lasttyperna enligt eget programs inställningar för att minska risken för felrapporteringar.

```
For J = 1 To RCC.Count
   AT = RCC.Get(J).AnalizeType
   If (AT = I_CAT_STATIC_LINEAR Or AT = I_CAT_STATIC_NONLINEAR) Then
      cr = "A" & Format(idx)
      Cells.Range(cr) = RCC.Get(J).Number
      cr = "B" & Format(idx)
      Cells.Range(cr) = RCC.Get(J).name
      cr = "C" & Format(idx)
      Cells.Range(cr) = RCC.Get(J).NatureName
      idx = idx + 1
   End If
Next J
```

Figur 5. Kod för uppgörande av lista för lastnummer, lastfall och lasttyp.

Importeringen och lagringen av lastfallen gavs namnet RCC i *Figur 4.* Koden i *Figur 5* räknar igenom samtliga lastfall och skriver ut uppgifter om lastfallsnummer, lastfallsnamn samt lasttyp och binder dessa till specifika celler. I detta fall läggs numret i kolumn A, namnet i kolumn B och typen i kolumn C. De tillgängliga lasttyperna skrivs ut enligt de första fyra raderna i koden ovan med början från cell (J+2;5) dvs. rad tre och framåt och kolumn fem (=E3…).

Med hjälp av koden ovan har man nu importerat befintliga lastfall i en öppnad Robot-modell. Nu är det fritt fram att modifiera dessa lastfall efter behov. Befintliga, importerade lastfall kan modifieras och nya lastfall kan skapas. En viktig begränsning som bör noteras är dock att det ännu inte går att radera befintliga lastfall. Vill man radera ett lastfall bör det göras direkt i Robot-modellen.

När informationen är modifierad skall knappen ”Uppdatera lastfall i Robot” användas för att, som namnet säger, uppdatera lastfallen i Robot-modellen. Nedan följer några utdrag ur koden för denna uppdatering.

```
If Not (loads.Exist(n)) Then

   loads.CreateSimple n, name, Nature, I_CAT_STATIC_LINEAR
   If tmpstr <> "" Then
      Set SC = loads.Get(n)
      SC.SetNatureExt Rglidx - 1
   End If

Else
   Set SC = loads.Get(n)
   SC.name = name
   SC.Nature = Nature
   If tmpstr <> "" Then SC.SetNatureExt Rglidx - 1
End If
idx = idx + 1
```

Figur 6. Kod för skapandet av nya lastfall.

Koden ovan illustrerad i *Figur 6* visar kommandona för hur skapandet av nya lastfall sker. Dessa kommandon finns inte med i API-guiden, utan hittades i ett färdigt program utsatt på Autodesks forum och användes som mall. Koden skapar nytt lastfall i fall det inte existerar och modifierar ett befintligt ifall lastfallsnumret är detsamma. Övriga delar av koden bygger på samma princip som importeringen, dock i omvänd ordning.

4.1.3.2 Lastfallkombinationer

Bakom den andra fliken ”Lastfallskombinationer” i *Figur 7* nedan, finner man programmet gjort som huvuddel i detta arbete.

Figur 7. Skärmdump över flikar använda i detta arbete framtagna Excel-program.

Ytterligare bakom detta program finner man en kod som gås igenom härefter. Denna kod kräver att det föregående programmet existerar eftersom lastfallskombinationer naturligt bygger på lastfall.

```
Dim CT(4) As String
CT(0) = "ULS"
CT(1) = "SLS"
CT(2) = "ACC"
CT(3) = "FIRE"

Dim RCCGP As RobotCodeCmbGenerationParams
Set RCCGP = RobApp.Project.Structure.Cases.CodeCmbEngine.Params

Sheet2.Range("A6", "AZ1000").ClearContents

For J = 1 To RCC.Count
   AT = RCC.Get(J).AnalizeType
   If (AT = I_CAT_COMB_NONLINEAR Or AT = I_CAT_COMB) Then
      Set RComb = RCC.Get(J)
      cr = "A" & Format(idy)
      Sheet2.Cells.Range(cr) = RComb.Number
      cr = "B" & Format(idy)
      Sheet2.Cells.Range(cr) = RComb.name
      cr = "C" & Format(idy)
      Sheet2.Cells.Range(cr) = CT(RComb.CombinationType)
      idy = idy + 1
   End If
Next J
```

Figur 8. Importering av kombinationsinformation.

Importeringen av kombinationer inleds som tidigare program med diverse aktiveringar och lagringar av information, vilket jag inte tagit med i koden ovan. Det som dock är nytt för denna kod och detta program är importeringen av kombinationstyp: brottgräns-, bruksgräns-, olycks- samt brandtillstånd. I de första fem raderna i *Figur 8* ovan finner man förkortningarna för dessa kombinationstillstånd som används i Robot i den ordning som uppräknats.

Innan kombinationerna gås igenom på samma vis som tidigare, raderas all tidigare information från cell A6 ända fram till AZ1000 för att undvika att gammal data lämnar och förvirrar användaren vid senare tillfälle. Därefter räknas kombinationsnummer, -namn samt -typ upp.

```
idymax = Sheet1.Cells(3, 7)
idy = 6

For J = 1 To RCC.Count
   AT = RCC.Get(J).AnalizeType

   If (AT = I_CAT_COMB_NONLINEAR Or AT = I_CAT_COMB) Then
      Set RComb = RCC.Get(J)
      For jj = 1 To RComb.CaseFactors.Count
         cn = RComb.CaseFactors.Get(jj).CaseNumber
         For ii = 3 To idymax + 3
            cr = "A" & Format(ii)
            If Sheet1.Range(cr).Value = cn Then
               colindx = 1 + ii
               Exit For
            End If
         Next ii
         Sheet2.Cells(idy, colindx) = RComb.CaseFactors.Get(jj).Factor
      Next jj
      idy = idy + 1
      Set RComb = Nothing
   End If
```

Figur 9. Importering av kombinationernas säkerhetskoefficienter.

Den första raden i koden i *Figur 9* är en hänvisning till cell G3 i flik ”Lastfall” i Excel-programmet. Kommandot är insatt för att räkna ut maxantalet lastfall som programmet skall gå igenom. Vidare plockas säkerhetskoefficienterna ut och radas upp i vågrät riktning i ark nummer två (Sheet2), medan programmet går igenom lastfallen i ark nummer ett (Sheet1). Övriga kommandon motsvarar till stor del det som tidigare gåtts igenom vid lastfallsimportering.

```
cr = "A" & Format(idx)
If IsEmpty(Cells.Range(cr).Value) Then
   Exit Do
End If
n = Cells.Range(cr).Value
cr = "B" & Format(idx)
name = Cells.Range(cr).Value
If n = 0 Then Exit Do
cr = "C" & Format(idx)
Dim ctn As String
ctn = Cells.Range(cr).Value
Dim CT As IRobotCombinationType
CT = I_CBT_ALS
If ctn = "uls" Or ctn = "ULS" Then
   CT = I_CBT_ULS
ElseIf ctn = "sls" Or ctn = "SLS" Then
   CT = I_CBT_SLS
ElseIf ctn = "acc" Or ctn = "ACC" Then
   CT = I_CBT_ALS
ElseIf ctn = "fire" Or ctn = "FIRE" Then
   CT = I_CBT_FIRE
End If
Dim cmb As IRobotCaseCombination
```

Figur 10. Uppdatering av kombinationsinformation i ROBOT.

Koden i *Figur 10* går igenom tidigare importerad information såsom nummer, namn och kombinationstyp, beroende på vilken cell de befinner sig i. Vid modifiering av kombinationer finns möjlighet att skriva in kombinationstypen med hjälp av både små och stora bokstäver.

Den resterande delen av uppdateringskoden kunde inte tas fram med hjälp av API-guiden, utan blev, med viss modifiering, kopierat från en mall skriven av Rafal Gaweda på Autodesk. Här var det fråga om mer ingående kunskaper av Robots källkod som inte den vanliga användaren har tillgång till, men som medlemmar på Autodesks forum får ta del av. Detta var alltså en väsentlig och mer eller mindre avgörande del för att mitt egna program skulle fungera.

Slutligen blev det nödvändigt att skapa en funktion för jämförelse av kombinationer med lastfall. Detta för att förhindra nummer- och namndubbletter. Koden jämför importerad

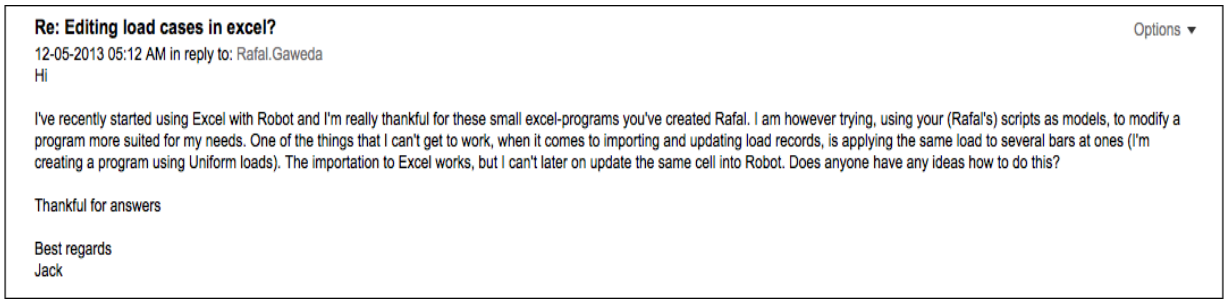
data före uppdatering och skickar ett felmeddelande ifall dubbletter upptäcks. Funktionen körs igenom varje gång endera kombinationerna eller lastfallen skall uppdateras.

4.1.3.3 Linjelaster

För att modifiera linjelaster i Robot skapades slutligen ett program med denna uppgift. Som utgångspunkt användes en mall för importering och uppdatering av ytlaster. Orsaken till att jag valde linjelaster framom ytlaster var att linjelasterna är lättare att applicera utan desto större precisering. Objektnummer utan koordinater räcker. Programmet blev dock mer en presentation av Excel-tillämpningens möjligheter än något man har konkret nytta av. Tanken med detta program är dock att kunna utveckla och förlänga funktionen så att t.ex. vindlaster och dylikt kan appliceras och modifieras i Robot-modellen.

4.1.3.4 Problemlösning

Programmeringen har bestått av utmaningar, idel problemlösning och svarssökning. Till all lycka finns Google att tillgå. Mestadels har dock Autodesks hjälpforum använts, vilket varit till stor hjälp. Dels finns färdiga trådar var folk haft motsvarande problem, dels har hjälp kunna efterfrågas. Nedan presenteras exempel på en fråga som ställt och de svar som erhållits.



Re: Editing load cases in excel?

Options ▾

12-05-2013 05:12 AM in reply to: Rafal.Gaweda

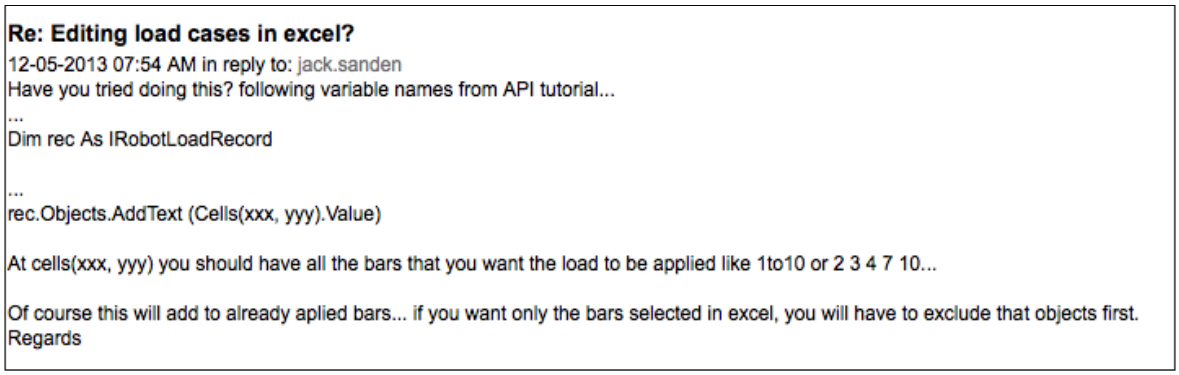
Hi

I've recently started using Excel with Robot and I'm really thankful for these small excel-programs you've created Rafal. I am however trying, using your (Rafal's) scripts as models, to modify a program more suited for my needs. One of the things that I can't get to work, when it comes to importing and updating load records, is applying the same load to several bars at ones (I'm creating a program using Uniform loads). The importation to Excel works, but I can't later on update the same cell into Robot. Does anyone have any ideas how to do this?

Thankful for answers

Best regards
Jack

Figur 11. Fråga som ställdes 05.12.2013 på Autodesks hjälpforum. Källa: forum.autodesks.com



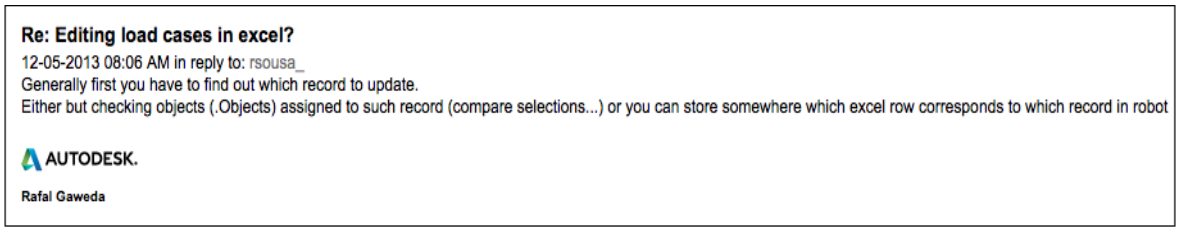
Re: Editing load cases in excel?
12-05-2013 07:54 AM in reply to: jack.sanden
Have you tried doing this? following variable names from API tutorial...
...
Dim rec As IRobotLoadRecord

...
rec.Objects.AddText (Cells(xxx, yyy).Value)

At cells(xxx, yyy) you should have all the bars that you want the load to be applied like 1to10 or 2 3 4 7 10...

Of course this will add to already aplied bars... if you want only the bars selected in excel, you will have to exclude that objects first.
Regards

Figur 12. Det första svaret som erhölls samma dag av användare "rsousa_". Källa: forums.autodesk.com



Re: Editing load cases in excel?
12-05-2013 08:06 AM in reply to: rsousa_
Generally first you have to find out which record to update.
Either but checking objects (.Objects) assigned to such record (compare selections...) or you can store somewhere which excel row corresponds to which record in robot

AUTODESK.

Rafal Gaweda

Figur 13. Svaret som API-guru Rafal Gaweda gav. Även det kom samma dag som frågan ställdes. Källa: forums.autodesk.com

Svaren som gavs var inte alltid de efterfrågade eller sökta, men de gav i regel en fingervisning om vari problemet låg. Att lära sig VB-kommandon och veta vad de gjorde krävde en hel del läsning på diverse VB-forum, var även svar hittades på somliga problem.

5. Resultat och tolkning

Avsikten med detta kapitel är att presentera de resultat som forskningen genererat. Forskningsfrågorna får utgöra den huvudsakliga utgångspunkten för resultatredovisningens struktur.

1. På vilket sätt kunde rutinmässiga uppgifter vid lastfallsberäkningar ersättas med hjälp av dataprogram?

Som ett resultat av forskningen noteras att det är tämligen enkelt att skapa användbara dataapplikationer som är användbara vid byggnadskonstruktion. Erfarenheten från detta forskningsprojekt visar att förhållandevis enkla datatillämpningar avsevärt kan underlätta annars tidsdryga rutinberäkningar och inmatningar. I huvudsak handlar arbetsprocessen i detta arbete om importering, modifiering och uppdatering av lastfallskombinationer. Dylika arbetsprocesser kan förenklas genom att skapa en brygga mellan exempelvis Excel och Robot. Teorin för denna typ av bryggor finns redan att tillgå för personer med intresse för programmering. På motsvarande sätt kan Excel givetvis kombineras med andra program.

Att hitta andra mer kreativa sätt som skulle rationalisera och underlätta arbetet, med manuella insatser, är inte sannolika. Resultatet indikerar också möjligheter att utvidga programmets användningspotentialer.

2. Vilka kvalitativa förbättringar kunde ett dylikt dataprogram bidra med?

Resultatet från detta projekt pekar på betydande potential att spara tid och därmed också kostnader. Samtidigt inbjuder projektet förhoppningsvis till innovativa tillämpningar inom både utbildningar och planeringsbyråer. Som ett kvalitativt resultat bör därför ses att projektet synliggör nya utvecklingspotentialer inom konstruktionsplanering.

I en vidare tolkning av resultaten bör noteras att denna programutveckling tillsvidare endast prövats teoretiskt. Därtill bör påpekas att resultatet och tillämpningen av detta arbete berör en avgränsad, men förvisso betydelsefull, del vid konstruktionsberäkning.

6. Kritisk granskning och diskussion

Oberoende av bransch, näringsliv eller offentlig sektor, är effektivare användning av tid och kostnadsbesparingar centrala drivkrafter i all verksamhet. Dock får detta inte ske på bekostnad av kvalitet eller säkerhet. Syftet med detta arbete har varit att presentera ett funktionellt dataprogram som kunde effektivera och underlätta byggnadskonstruktörens arbete, i detta fall begränsat till lastfallskombinationer. Granskningen av de diskussioner som sker inom internationella nätforum visar på behov och intresse för dylika lösningar.

Av den teoretiska referensramen framgår att inom konstruktionsbranschen finns många delområden som kunde automatiseras och effektiveras genom att tillvarata de möjligheter som exempelvis Excel och Robot erbjuder. Jag vill här understryka att många program var för sig är effektiva, men mervärde kan uppnås genom kreativ kombination av programfunktioner.

Den teoretiska och praktiska nytta som detta program kan erbjuda är, förutom tids- och kostnadsinbesparingar, framför allt inspiration till vidareutveckling av motsvarande applikationer.

Resultaten från detta forskningsprojekt visar entydigt att förhållandevis enkla dataprogram effektiv kunde erbjuda analyshjälp vid framtagning av lastfallkombinationer. Dock behöver understrykas att det dataprogram som framtagits inom ramen för detta forskningsprojekt inte prövats eller utvärderats i en autentisk miljö av andra oberoende konstruktörer. Därmed kan inga långtgående slutsatser i detta skede dras beträffande verkliga tidsinbesparingar eller kostnadsvinningar.

Som svaghet kan framhållas att programmet ännu inte har en användarvänlig layout. För att ytterligare förbättra programmets funktion krävs mera ingående kunskap i programmering.

Avslutningsvis noteras nyttan av programmeringsfärdigheter hos konstruktörer idag och framledes. Det är rimligt att anta att vi inom de närmaste åren kommer att få se flera innovationer på detta område. Programvarorna kommer fortsättningsvis ständigt att förbättras, men konkurrensfördelarna kommer att uppnås av de som kreativt kan importera, modifiera och uppdatera effektivt mellan dessa.

7. Källförteckning

Autodesk Inc. (2010). *API Tutorial*. http://forums.autodesk.com/t5/Robot-Structural-Analysis/API-Tutorial/td-p/3939601 (hämtat: 25.11.2013).

Mabbutt, D. (u.å.). *About Visual Basic and About This Site*. http://visualbasic.about.com/od/vbnetspecialtopics/a/aboutvb.htm (hämtat: 17.01.2014).

Microsoft Developer Network. http://msdn.microsoft.com/en-us/library/ (hämtat: 18.02.2014)

Nilsson, B. (2013). *Finita elementmetoden - En kort introduktion till teorin*. Halmstad: Högskolan i Halmstad.

RIL-201-1-2011. (2011). *Suunnitteluperusteet ja rakenteiden kuormat*. Suomen Rakennusinsinöörien Liitto ry.

Visual Basic (2014). http://sv.wikipedia.org/wiki/Visual_Basic (hämtat: 28.02.2014)

Eurocode-standarder:

Nationell Bilaga. Till standard SFS-EN 1990 Eurokod. Dimensioneringsgrunder för bärande konstruktioner (2007). http://www.ym.fi/download/noname/{803140DC-DFC2-4638-B351-127069DBF3B9}/32700 (hämtat: 02.04.2014)

SS-EN 1990 (2002). *Grundläggande dimensioneringsregler för bärverk.* Svensk version. ICS 91.010.30. Utgåva nr 1. European Committee for Standardization.

BILAGA 1

```
Private Sub Uppdatera_lastfall_i_Robot_Click()

If MsgBox("Är du säker på att du vill uppdatera?", vbYesNo) = vbNo Then Exit Sub

Dim n As Long
Dim name As String
Dim RobApp As IRobotApplication
Set RobApp = New RobotApplication
Dim loads As IRobotCaseServer
Dim SC As RobotSimpleCase
Dim Nature As Integer
Dim tmpstr As String
If Not CaseCheck Then Exit Sub

Set loads = RobApp.Project.Structure.Cases

Dim RCCGP As RobotCodeCmbGenerationParams
Set RCCGP = RobApp.Project.Structure.Cases.CodeCmbEngine.Params

Dim Names(6) As String

Set RCase = RobApp.Project.Structure.Cases.CreateSimple(100000, "tmp", 0, I_CAT_STATIC_LINEAR)
For i = 0 To 6
    RCase.Nature = i
    Names(i) = RCase.NatureName
Next i


RobApp.Project.Structure.Cases.Delete (100000)
Dim SubNatures() As String
Dim Subnames() As String
Dim NumberofNatures As Integer

NumberofNatures = RCCGP.Regulations.Actions.Count

ReDim SubNatures(NumberofNatures)
ReDim Subnames(NumberofNatures)

For J = 1 To NumberofNatures
    Subnames(J) = RCCGP.Regulations.Actions.GetName(J)
    If (Subnames(J) = "") Then Subnames(J) = Names(RCCGP.Regulations.Actions.GetNature(J))
    SubNatures(J) = RCCGP.Regulations.Actions.GetNature(J)
Next J

idx = 3
Dim Rglidx As Integer
Do
    cr = "A" & Format(idx)
    n = Cells.Range(cr).Value
    cr = "B" & Format(idx)
    name = Cells.Range(cr).Value
    If n = 0 Then Exit Do

    tmpstr = LCase(Cells.Range("C" & Format(idx)).Value)
       Rglidx = 0


       For i = 1 To NumberofNatures
        If tmpstr = LCase(Subnames(i)) Then
            Nature = SubNatures(i)
            Rglidx = i
            Exit For
        End If
       Next i

    If (Rglidx = 0) Then
            MsgBox "Hittar ingen belastning (subnature) med namnet:  " & tmpstr & vbNewLine + _
            "Var vänlig och kontrollera:" + vbNewLine + "1. Stavning" + vbNewLine + "2. Att du använt någo
            Exit Sub
        End If
```

```
    If Not (loads.Exist(n)) Then

        loads.CreateSimple n, name, Nature, I_CAT_STATIC_LINEAR
        If tmpstr <> "" Then
            Set SC = loads.Get(n)
            SC.SetNatureExt Rglidx - 1
        End If

    Else
        Set SC = loads.Get(n)
        SC.name = name
        SC.Nature = Nature
        If tmpstr <> "" Then SC.SetNatureExt Rglidx - 1
    End If
    idx = idx + 1
Loop
End Sub

Private Sub Importera_lastfall_Click()

If MsgBox("Är du säker på att du vill importera?", vbYesNo) = vbNo Then Exit Sub

Dim RobApp As IRobotApplication
Set RobApp = New RobotApplication

If Not RobApp.Visible Then
    Set RobApp = Nothing
    MsgBox "Var vänlig och öppna en modell i Robot först!", vbOKOnly
    Exit Sub
End If


Dim RCC As RobotCaseCollection
Set RCC = RobApp.Project.Structure.Cases.GetAll
idx = 3

Range("A3", "E100").ClearContents

Dim Names(6) As String

[E1].Value = "Tillgängliga belastningar"

Set RCase = RobApp.Project.Structure.Cases.CreateSimple(100000, "tmp", 0, I_CAT_STATIC_LINEAR)
For i = 0 To 6
    RCase.Nature = i
    Names(i) = RCase.NatureName
Next i


RobApp.Project.Structure.Cases.Delete (100000)

Dim RCCGP As RobotCodeCmbGenerationParams
Set RCCGP = RobApp.Project.Structure.Cases.CodeCmbEngine.Params


For J = 1 To RCCGP.Regulations.Actions.Count
    Cells(J + 2, 5) = RCCGP.Regulations.Actions.GetName(J)
    If (RCCGP.Regulations.Actions.GetName(J) = "") Then Cells(J + 2, 5) = Names(RCCGP.Regulations.Actions.GetNature(J))
Next J

For J = 1 To RCC.Count
    AT = RCC.Get(J).AnalizeType
    If (AT = I_CAT_STATIC_LINEAR Or AT = I_CAT_STATIC_NONLINEAR) Then
        cr = "A" & Format(idx)
        Cells.Range(cr) = RCC.Get(J).Number
        cr = "B" & Format(idx)
        Cells.Range(cr) = RCC.Get(J).name
        cr = "C" & Format(idx)
        Cells.Range(cr) = RCC.Get(J).NatureName
        idx = idx + 1
    End If
```

```
Next J


End Sub
```

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)

End Sub
```

```
Function CaseCheck() As Boolean

CaseCheck = True

idx = 3
Do
    cr = "A" & Format(idx)
    n = Sheet1.Cells.Range(cr).Value
    If n = 0 Then Exit Do

    idy = 6
    Do
        cy = "A" & Format(idy)
        ny = Sheet2.Cells.Range(cy).Value
        If ny = 0 Then Exit Do
        If n = ny Then
            MsgBox "Lastfall/kombination nummer" & Str(n) & " finns redan!", vbCritical, "ERROR"
            CaseCheck = False
            Exit Function
        End If
        idy = idy + 1
    Loop
    idx = idx + 1
Loop

End Function
```

BILAGA 2

```
Private Sub Importera_kombinationer_Click()

If MsgBox("Är du säker på att du vill importera?", vbYesNo, "Observera!") = vbNo Then Exit Sub

Dim RobApp As IRobotApplication
Set RobApp = New RobotApplication

If Not RobApp.Visible Then
    Set RobApp = Nothing
    MsgBox "Var vänlig och öppna en modell i Robot först!", vbOKOnly, "Ett fel uppstod!"
    Exit Sub
End If

Dim RCC As RobotCaseCollection
Dim RComb As IRobotCaseCombination
Set RCC = RobApp.Project.Structure.Cases.GetAll

idx = 3
idy = 6

Dim CT(4) As String
CT(0) = "ULS"
CT(1) = "SLS"
CT(2) = "ACC"
CT(3) = "FIRE"

Dim RCCGP As RobotCodeCmbGenerationParams
Set RCCGP = RobApp.Project.Structure.Cases.CodeCmbEngine.Params

Sheet2.Range("A6", "AZ1000").ClearContents

For J = 1 To RCC.Count
    AT = RCC.Get(J).AnalizeType
    If (AT = I_CAT_COMB_NONLINEAR Or AT = I_CAT_COMB) Then
        Set RComb = RCC.Get(J)
        cr = "A" & Format(idy)
        Sheet2.Cells.Range(cr) = RComb.Number
        cr = "B" & Format(idy)
        Sheet2.Cells.Range(cr) = RComb.name
        cr = "C" & Format(idy)
        Sheet2.Cells.Range(cr) = CT(RComb.CombinationType)
        idy = idy + 1
    End If
Next J

If idy = 6 Then MsgBox "Kunde inte hitta någon lastfallskombination i modellen!", vbOKOnly, "Ett fel uppstod!"

idymax = Sheet1.Cells(3, 7)
idy = 6

For J = 1 To RCC.Count
    AT = RCC.Get(J).AnalizeType

    If (AT = I_CAT_COMB_NONLINEAR Or AT = I_CAT_COMB) Then
        Set RComb = RCC.Get(J)
        For jj = 1 To RComb.CaseFactors.Count
            cn = RComb.CaseFactors.Get(jj).CaseNumber
            For ii = 3 To idymax + 3
                cr = "A" & Format(ii)
                If Sheet1.Range(cr).Value = cn Then
                    colindx = 1 + ii
                    Exit For
                End If
```

```
            Next ii
            Sheet2.Cells(idy, colindx) = RComb.CaseFactors.Get(jj).Factor
        Next jj
        idy = idy + 1
        Set RComb = Nothing
    End If

Next J

End Sub
```

```
Private Sub Uppd_komb_i_Robot_Click()

If MsgBox("Är du säker på att du vill uppdatera?", vbYesNo, "Observera!") = vbNo Then Exit Sub

Dim RobApp As IRobotApplication
Set RobApp = New RobotApplication

If Not RobApp.Visible Then
    Set RobApp = Nothing
    MsgBox "Var vänlig och öppna en modell i Robot först!", vbOKOnly, "Ett fel uppstod!"
    Exit Sub
End If

Dim n As Long
Dim name As String
Dim loads As IRobotCaseServer

If Not Sheet1.CaseCheck Then Exit Sub

Set loads = RobApp.Project.Structure.Cases
idx = 6
Do
 Cells(2, 24) = Str(idx - 5) & " / " & (Cells(20000, 1).End(xlUp).row - 6)

 cr = "A" & Format(idx)
 If IsEmpty(Cells.Range(cr).Value) Then
     Exit Do
 End If
 n = Cells.Range(cr).Value
 cr = "B" & Format(idx)
 name = Cells.Range(cr).Value
 If n = 0 Then Exit Do
 cr = "C" & Format(idx)
 Dim ctn As String
 ctn = Cells.Range(cr).Value
 Dim CT As IRobotCombinationType
 CT = I_CBT_ALS
 If ctn = "uls" Or ctn = "ULS" Then
     CT = I_CBT_ULS
 ElseIf ctn = "sls" Or ctn = "SLS" Then
     CT = I_CBT_SLS
 ElseIf ctn = "acc" Or ctn = "ACC" Then
     CT = I_CBT_ALS
 ElseIf ctn = "fire" Or ctn = "FIRE" Then
     CT = I_CBT_FIRE
 End If
 Dim cmb As IRobotCaseCombination

 If loads.Exist(n) Then
     Set cmb = loads.Get(n)
     cmb.name = name
     cmb.CombinationType = CT
 Else
     Set cmb = loads.CreateCombination(n, name, CT, I_CN_EXPLOATATION, I_CAT_COMB)
 End If
```

```
Dim SC As Long
Dim sc_row As Range, row As Range
Dim cell As Range
Set sc_row = Sheet2.Rows(4)
Set row = Sheet2.Rows(idx)
Dim i As Integer
For i = 4 To sc_row.Cells.Count
    Set cell = sc_row.Cells(i)
    If IsEmpty(cell.Value) Then
       Exit For
    End If
    SC = cell.Value

    If SC <> 0 Then

    If cmb.CaseFactors.Count <> 0 Then
    ik = 0
    Do
          ik = ik + 1
          If cmb.CaseFactors.Get(ik).CaseNumber = SC Then
              cmb.CaseFactors.Delete (ik)
              If (ik <> 0) Then ik = ik - 1
          End If
      Loop While (ik < cmb.CaseFactors.Count)
      End If

      Set cell = row.Cells(i)
      If Not IsEmpty(cell.Value) Then

          If Not loads.Exist(SC) Then
             MsgBox "Lastfall " & Str(SC) & " finns ännu inte och kan" + vbNewLine + _
             "därmed inte användas i kombination " & Str(n) & "." + vbNewLine + "Var vänlig och uppdatera lastfa
             Exit Sub
          End If

          Dim f As Double
          f = cell.Value
          If f <> 0 Then
              cmb.CaseFactors.New SC, f
          End If
      End If
      Else
          If i > 17 Then Exit For
      End If
   Next i
   idx = idx + 1
Loop
Cells(2, 24) = ""
End Sub
```

BILAGA 3

```
Private Sub Importera_linjelaster_Click()
Range("A10", "H500").ClearContents
Set RobApp = New RobotApplication
Dim loads As RobotCaseServer

If Not RobApp.Visible Then
    Set RobApp = Nothing
    MsgBox "Var vänlig och öppna en modell i Robot först!", vbOKOnly, "Ett fel uppstod!"
    Exit Sub
End If

Set loads = RobApp.Project.Structure.Cases

    If Range("b3").Value <> Empty And Int(Range("b3").Value) <> 0 Then
        CaseNumber = Int(Range("b3").Value)
    Else
        MsgBox "Var vänlig och skriv in önskat lastfallsnummer i cell B3", vbOKOnly, "Observera!"
        Exit Sub
    End If

Set loads = RobApp.Project.Structure.Cases

If Not loads.Exist(CaseNumber) Then
    MsgBox "Lastfall nummer" & Str(CaseNumber) & " kan inte hittas i modellen!" + vbNewLine + vbNewLine +
    Exit Sub
End If

If loads.Get(CaseNumber).Type <> I_CT_SIMPLE Then
    MsgBox "Lastfall nummer" & Str(CaseNumber) & " är en lastfallskobination!" + vbNewLine + vbNewLine +
    Exit Sub
End If

Dim Cas As RobotSimpleCase
Dim Recs As RobotLoadRecordMngr
Dim Rec_1 As RobotLoadRecord

     Set Cas = loads.Get(CaseNumber)
     Set Recs = Cas.Records

 Dim GLO(2) As String
 Dim Projected(2) As String

 GLO(0) = "global"
 GLO(1) = "local"
 Projected(0) = "not projected"
 Projected(1) = "projected"


     idx = 10
     For i = 1 To Recs.Count

     If Recs.Get(i).Type = I_LRT_BAR_UNIFORM Then

         Set Rec_1 = Recs.Get(i)
         Cells(idx, 1) = i
         Cells(idx, 2) = "Uniform Load"
         Cells(idx, 3) = Rec_1.Objects.ToText
         Cells(idx, 4) = Rec_1.GetValue(I_BURV_PX) / 1000
         Cells(idx, 5) = Rec_1.GetValue(I_BURV_PY) / 1000
         Cells(idx, 6) = Rec_1.GetValue(I_BURV_PZ) / 1000
         Cells(idx, 7) = GLO(Rec_1.GetValue(I_URV_LOCAL_SYSTEM))
         Cells(idx, 8) = Projected(Rec_1.GetValue(I_URV_PROJECTED))

         idx = idx + 1
     End If
     Next i
     If idx = 10 Then MsgBox "Kunde inte hitta någon linjelast i lastfall nummer " & CaseNumber

     Set Recs = Nothing
     Set Rec_1 = Nothing
     Set Cas = Nothing
End Sub
```

```
Private Sub Uppd_linjelast_i_Robot_Click()

If MsgBox("Är du säker på att du vill uppdatera linjelasterna?", vbYesNo) = vbNo Then Exit Sub

On Error GoTo err

Set RobApp = New RobotApplication
Dim loads As RobotCaseServer

If Not RobApp.Visible Then
    Set RobApp = Nothing
    MsgBox "Var vänlig och öppna en modell i Robot först!", vbOKOnly, "Ett fel uppstod!"
    Exit Sub
End If

Set loads = RobApp.Project.Structure.Cases

    If Range("b3").Value <> Empty And Int(Range("b3").Value) <> 0 Then
        CaseNumber = Int(Range("b3").Value)
    Else
        MsgBox "Var vänlig och skriv in önskat lastfallsnummer i cell B3", vbOKOnly, "Observera!"
        Exit Sub
    End If

    Set loads = RobApp.Project.Structure.Cases

    If Not loads.Exist(CaseNumber) Then
        MsgBox "Lastfall nummer" & Str(CaseNumber) & " kan inte hittas i modellen!", vbOKOnly, "Ett fel uppst
        Exit Sub
    End If

    If loads.Get(CaseNumber).Type <> I_CT_SIMPLE Then
        MsgBox "Case" & Str(CaseNumber) & " is not a simple case", vbOKOnly, "ERROR"
        Exit Sub
    End If

    Dim Cas As RobotSimpleCase
    Dim Recs As RobotLoadRecordMngr
    Dim Rec_1 As RobotLoadRecord
    Set Cas = loads.Get(CaseNumber)
    Set Recs = Cas.Records

Dim GLO As String
Dim Projected As String

    idx = 10
While Not IsEmpty(Cells(idx, 1))
    i = Int(Cells(idx, 1))
    If Recs.Get(i).Type = I_LRT_BAR_UNIFORM Then

        Set Rec_1 = Recs.Get(i)

        Rec_1.Objects.FromText (Cells(idx, 3).Value)
        Rec_1.SetValue I_BURV_PX, CDbl(Cells(idx, 4)) * 1000
        Rec_1.SetValue I_BURV_PY, CDbl(Cells(idx, 5)) * 1000
        Rec_1.SetValue I_BURV_PZ, CDbl(Cells(idx, 6)) * 1000

        GLO = LCase(Cells(idx, 7).Value)
        If GLO = "global" Then
            Rec_1.SetValue I_URV_LOCAL_SYSTEM, 0
        Else
            Rec_1.SetValue I_URV_LOCAL_SYSTEM, 1
        End If
```

```
        Projected = LCase(Cells(idx, 8).Value)
        If Projected = "projected" Then
            Rec_1.SetValue I_URV_PROJECTED, 1
        Else
            Rec_1.SetValue I_URV_PROJECTED, 0

        End If
        idx = idx + 1
    End If
Wend

    Set Recs = Nothing
    Set Rec_1 = Nothing
    Set Cas = Nothing
    Exit Sub
err:
MsgBox "Ett fel har uppstått!"
End Sub
```