MatrixSolver

Lucas Arnström & Lukas Wirne & Oscar Wallster

6 mars 2014

Innehåll

1 Inledning

En matris är ett rektangulärt schema av element, där varje element representerar data i form av tal. Såhär kan en matris se ut.

$$MatrisA = \begin{pmatrix} 1_{(1,1)} & 2_{(1,2)} & 3_{(1,3)} \\ 4_{(2,1)} & 5_{(2,2)} & 6_{(2,3)} \\ 7_{(3,1)} & 8_{(3,2)} & 9_{(3,3)} \end{pmatrix}$$

Just för detta exempel visas positionen för varje element i A inom parenteserna. För elementet med värde 6, har positionen (2,3), vilket betyder rad 2 och kolumn 3. Matrisen ovan är en 3·3 matris. Den har tre rader, och tre kolumner. En matris behöver inte ha lika många rader som kolumner. Matriser används i stor utsträckning inom matematiken och fysiken, de underlättar beräkningarna för tredimensionella kroppar, till exempel för de som arbetar med grafik, eller hållfasthetslära. Beräkningarna som används på matriser är många, och de är inte alltid så enkla att göra för hand. En dator kan göra beräkningarna på matriser både enklare och snabbare, speciellt för matriser med stora dimensioner. I denna rapport kommer ett program presenteras som kan utföra dessa beräkningar.

2 Sammanfattning

MatrixSolver är ett program som hjälper användaren att göra beräkningar på matriser. Programmet kan utföra addition, subtraktion, och multiplikation på två matriser. Den kan också hitta invers, determinant, kofaktormatris och adjugat till en matris. MatrixSolver underlättar arbetet för alla som utför matrisberäkningar för hand. Programmet används via en terminal, och består utav utav en TUI (terminal user interface).

2.1 Användningsmanual

För att starta programmet så kallar användaren på funktionen ”start()” i filen ”main.sml”. Den kommer då att be användaren om att mata in en matris. För att mata in en matris ska den skrivas på följande vis, här är ett exempel.

$$\begin{pmatrix} 1 & 2 & \frac{3}{4} \\ 4 & 5 & 6 \\ 7 & \frac{8}{3} & 9 \end{pmatrix}$$

För att mata in en matris i programmet så ramas den först in i ”{” och ”}” som markerar start och slut på matrisen. Därefter läggs varje rad i matrisen inom dessa ramar. Dessa ramas in även de på ett likadant sätt för att markera start och slut på raden. Raderna separeras sedan med ett kommatecken. Mellanslag i inmatningen är ej nödvändigt men kan underlätta för användaren att utläsa vad denne matat in. Varje tal i matrisen matas in som ett bråktal, även heltalen. Varje tal i varje rad separeras även de med ett kommatecken.

Det vill säga, matrisen ovan matas in som:

$$\{\{1/1, 2/1, 3/4\}, \{4/1, 5/1, 6/1\}, \{7/1, 8/3, 9/1\}\}$$

Efter att användaren matat in den första matrisen i programmet så kommer den genast att be användaren om en andra matris. Det är inte nödvändigt att mata in en

till matris. För att hoppa över detta steg så kan användaren mata in ”C” som avbryter inmatningen och går vidare till nästa steg.

Beroende på om användaren matar in en eller två matriser så får denne upp olika alternativ. Väljer användaren att endast mata in en matris så får denna upp ett fyra alternativ, vilka är inversen, determinanten, adjugatet, och kofaktormatrisen utifrån den matris som matats in. Varje val görs genom att mata in den siffra som står framför varje alternativ i listan som dyker upp.

Skulle dock användaren väljat att mata in två matriser så får denne då tre helt andra alternativ att välja mellan. Dessa är addition, subtraktion, och multiplikation mellan de två matriserna. Precis som innan så görs även här varje val genom att mata in siffran som står framför varje alternativ.

När en uträkningen har valts så utför programmet denna uträkning och skriver sedan ut resultat för användaren. Därefter är programmet slut. Skulle användaren vilja köra det igen måste denne återigen anropa funktionen ”start()”.

3 Programdokumentation

3.1 Abstrakta datatyper

3.1.1 Fractal

Den abstrakta datatypen ”fractal” representerar ett bråk i programmet. Det finns tre alternativ för att skapa ett nytt bråktal, ett utav dem är att kalla på funktionen ”toFractal(<heltal>)” som resulterar i bråktalet $\frac{<\text{heltal}>}{1}$, ett annat är att kalla på funktionen ”createFractal(<täljare>, <nämnare >)” som ger resultatet $\frac{<\text{täljare}>}{<\text{nämnare}>}$, och det sista är att kalla på funktionen ”fractalFromString(<sträng>)”. Den sistnämnda funktionen kräver en korrekt strukturerad sträng som argument för att den ska fungera korrekt. Ett exempel på en korrekt sådan är ”1/4” som representerar en fjärdedel. För att strängen ska vara korrekt krävs det tre delar, en heltalstäljare, ett snedsträck som skiljer täljaren och nämnaren åt, och en heltalsnämnare. Det får inte finnas några mellanslag i strängen eller några andra tecken utöver de som nämnts.

Varje gång en funktion som gör en ändring på ett bråktal anropas så blir resultatet automatiskt förenklat. Detta ser till att bråket hela tiden är i sin enklaste form under programmets exekvering och underlättar läsbarheten när talet sedan ska skrivas ut för användaren. De funktioner som gör faktiska ändringar utav ett bråktal är ”fracOp”, ”fracAdd”, ”fracSub”, ”fracMult”, och ”fracDivide”.

3.1.2 Matrix

Den abstrakta datatypen ”matrix” representerar en matris med bråktal i programmet. Det är denna datatyp som innehåller de viktigaste funktionerna som gör de flesta beräkningarna i programmet. Det finns två alternativ för att skapa en ny matris. Ett utav de är att kalla på funktionen ”createMatrix(<2D-bråklista>)” som skapar en ny matris utifrån argumentet. Det andra alternativet är att kalla på funktionen ”parseMatrix(<sträng>)” som tar emot en sträng och returnerar en matris. Här är det väldigt viktigt att strängen är korrekt formulerad för att funktionen ska fungera som den ska. För regler gällande struktur på strängen var god se rubrik ”2.1 Användningsmanual”.

3.2 Programmets algoritmer

3.2.1 mAdd

mAdd(m1, m2)
TYPE: matrix * matrix → matrix
PRE: True
POST: The result of the two matrixes m1 and m2 added to eachother.

Funktionen är baserad på algoritmen för elementvis addition. För att addition mellan två matriser ska kunna utföras så måste matriserna ha samma dimensioner. Om A och B är två matriser, så finns en matris C så att A + B = C. Varje element i C med en position (x,y) skapas genom att addera elementet med position (x,y) från A med elementet med position (x,y) från B.

Exempel:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} + \begin{pmatrix} 3 & 2 & 3 \\ 11 & 5 & 56 \\ 2 & 0 & 91 \end{pmatrix} = \begin{pmatrix} (1+3) & (2+2) & (3+3) \\ (4+11) & (5+5) & (6+56) \\ (7+2) & (8+0) & (9+91) \end{pmatrix}$$

3.2.2 mSub

mSub(m1, m2)
TYPE: matrix * matrix → matrix
PRE: True
POST: The result of the two matrixes m1 and m2 subtracted from eachother.

Funktionen är baserad på algoritmen för elementvis subtraktion. För att subtraktion mellan två matriser ska kunna utföras så måste matriserna ha samma dimensioner. Om A och B är två matriser, så finns en matris C så att A - B = C. Varje element i C med en position (x, y) skapas genom att subtrahera elementet med position (x, y) från A med elementet med position (x, y) från B.

Exempel:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} - \begin{pmatrix} 3 & 2 & 3 \\ 11 & 5 & 56 \\ 2 & 0 & 91 \end{pmatrix} = \begin{pmatrix} (1-3) & (2-2) & (3-3) \\ (4-11) & (5-5) & (6-56) \\ (7-2) & (8-0) & (9-91) \end{pmatrix}$$

3.2.3 mMult

mMult(m1, m2)
TYPE: matrix * matrix → matrix
PRE: True
POST: The result of the two matrixes m1 and m2 multiplied with eachother.

Funktionen är baserad på algoritmen för elementvis multiplikation. För att multiplikation mellan matriserna A och B ska kunna utföras, så måste antalet kolumner i A vara desamma som antalet rader i B. Om C = A·B, så kan ett element från C med

position (x, y) bestämmas genom att multiplicera ihop första elementet från rad x i A, med första elementet i kolumn y i B, sedan addera detta med andra elementet från rad x i A, med andra elementet i kolumn y i B, detta görs ända tills alla element i rad x och kolumn y är adderade med varandra. Antal rader i C kommer bli samma som antal rader i A, och antal kolumner i C kommer bli samma som antal kolumner i B. När detta implementerades i funktionen så gjordes en transponat av B (kolumn 1 görs om till rad 1, kolumn 2 görs om till rad 2, osv) för att enklare kunna multiplicera och addera elementen.

Exempel:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \cdot \begin{pmatrix} 3 & 2 \\ 1 & 5 \\ 2 & 0 \end{pmatrix} = \begin{pmatrix} (1 \cdot 3 + 2 \cdot 1 + 3 \cdot 2) & (1 \cdot 2 + 2 \cdot 5 + 3 \cdot 0) \\ (4 \cdot 3 + 5 \cdot 1 + 6 \cdot 2) & (4 \cdot 2 + 5 \cdot 5 + 6 \cdot 0) \end{pmatrix}$$

3.2.4 flipp

flipp l
TYPE: 'a list list -> 'a list list
PRE: True
POST: Each column of elements in l has become separate rows.

Denna funktion beräknar transponaten av en matris. för att få fram transponaten av en matris m, görs alla rader om till kolumner, och kolumner om till rader. Första raden blir till första kolumnen, och första kolumnen blir till första raden.
Exempel:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 0 & 4 \\ 9 & 3 & 4 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix}$$

3.2.5 mDet

mDet m
TYPE: matrix $\rightarrow$ fractal
PRE: Matrix m is a non-empty square matrix.
POST: Fractal corresponding to the determinant of the matrix m.

Funktionen är baserad på algoritmen laplaceutveckling, som beräknar determinanten för en matris A genom att dela upp matrisen i mindre matriser, ända tills matriserna är av storlek 1·1. En matris med endast ett element, kommer få värdet på elementet som determinant. Nedan visas ett exempel där determinanten räknas ut för en 3·3 matris med hjälp av laplaceutveckling.

Steg 1. Multiplicera varannat element från första raden i matrisen med (-1), börja med element nummer 2.

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 0 & 4 \\ 9 & 3 & 4 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & -2 & 3 \\ 4 & 0 & 4 \\ 9 & 3 & 4 \end{pmatrix}$$

Steg 2. Nu ska Varje element från första raden multipliceras med tillhörande matris, sedan ska allt adderas ihop. Tillhörande matris får du ut genom att ta bort den rad och kolumn som elementet ligger i från hela matrisen.

$$1 \cdot \begin{pmatrix} 0 & 4 \\ 3 & 4 \end{pmatrix} - 2 \cdot \begin{pmatrix} 4 & 4 \\ 9 & 4 \end{pmatrix} + 3 \cdot \begin{pmatrix} 4 & 0 \\ 9 & 3 \end{pmatrix}$$

Steg 3. repetera steg 1 och 2 tills alla matriser är av dimension $1 \cdot 1$. För exempelmatrisen behöver stegen göras en gång till, nedan visas resultatet.

$$1 \cdot (0 \cdot (4) - 4 \cdot (3)) - 2 \cdot (4 \cdot (4) - 4 \cdot (9)) + 3 \cdot (4 \cdot (3) - 0 \cdot (3)) = 64$$

3.2.6 mCofactor

mCofactor (m)
TYPE: Matrix $->$ Matrix
PRE: Square matrix(nXn)
POST: Returns the cofactor matrix of m

Funktionen tar fram en kofaktormatris av en matris. Metoden som används är att ta fram alla minorer genom att göra mindre matriser av orginalmatrisen. Alla element i matrisen byts ut till determinanten av elementets tillhörande matris. Den tillhörande matrisen till ett element e, är originalmatrisen där rad och kolumn för e är borttagna. Multiplicera sedan -1 med de element som befinner sig på en jämn rad och ojämn kolumn, eller jämn kolum och ojämn rad.

Exempel:

$$\begin{pmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 1 & 0 & 6 \end{pmatrix} \rightarrow \begin{pmatrix} \begin{vmatrix} 4 & 5 \\ 0 & 6 \end{vmatrix} & \begin{vmatrix} 0 & 5 \\ 1 & 6 \end{vmatrix} & \begin{vmatrix} 0 & 4 \\ 1 & 0 \end{vmatrix} \\ \begin{vmatrix} 2 & 3 \\ 0 & 6 \end{vmatrix} & \begin{vmatrix} 1 & 3 \\ 1 & 6 \end{vmatrix} & \begin{vmatrix} 1 & 2 \\ 1 & 0 \end{vmatrix} \\ \begin{vmatrix} 2 & 3 \\ 4 & 5 \end{vmatrix} & \begin{vmatrix} 1 & 3 \\ 0 & 5 \end{vmatrix} & \begin{vmatrix} 1 & 2 \\ 0 & 4 \end{vmatrix} \end{pmatrix} \rightarrow \begin{pmatrix} 24 & -5 & -4 \\ 12 & 3 & -2 \\ -2 & 5 & 4 \end{pmatrix} \rightarrow \begin{pmatrix} 24 & 5 & -4 \\ -12 & 3 & 2 \\ -2 & -5 & 4 \end{pmatrix}$$

3.2.7 mAdjoint

mAdjoint m
TYPE: Matrix $\rightarrow$ Matrix
PRE: squared matrix(nXn)
POST: Returns the adjoint of matrix m

Adjugatet av en matris används för att ta fram inversen på en matris. Funktionen använder sig av funktionerna mCofactor och flipp(transponeringsfunktionen). Adjugatet av en matris m, är transponaten av kofaktormatrisen av m. Exempel:

$$m = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 1 & 0 & 6 \end{pmatrix} \rightarrow Cofactor(m) = \begin{pmatrix} 24 & 5 & -4 \\ -12 & 3 & 2 \\ -2 & -5 & 4 \end{pmatrix} \rightarrow Adjoint(m) = \begin{pmatrix} 24 & -12 & -2 \\ 5 & 3 & -5 \\ -4 & 2 & 4 \end{pmatrix}$$

3.2.8 mInv

mInv m
TYPE: matrix → matrix
PRE: The determinant of the matrix m is not equal to zero.
POST: Matrix corresponding to the inverse of the matrix m.

Funktionen räknar ut inversen av en matris m. Som precondition visar kan inte inversen beräknas på en tom matris. För att beräkna inversen beräknas determinanten d av matrisen m och adjugatet a från matris m, sedan används formeln: Invers av m = $(\frac{1}{d}) \cdot a$.

Exempel:

$$m = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 1 & 0 & 6 \end{pmatrix}$$

Determinanten av m = 22

$$Adjoint(m) = \begin{pmatrix} 24 & -12 & -2 \\ 5 & 3 & -5 \\ -4 & 2 & 4 \end{pmatrix}$$

$$\text{Invers av m} = m^{-1} = \frac{1}{22} \cdot \begin{pmatrix} 24 & -12 & -2 \\ 5 & 3 & -5 \\ -4 & 2 & 4 \end{pmatrix} = \begin{pmatrix} \frac{12}{11} & \frac{-6}{11} & \frac{-1}{11} \\ \frac{5}{22} & \frac{3}{22} & \frac{-5}{22} \\ \frac{-2}{11} & \frac{1}{11} & \frac{2}{11} \end{pmatrix}$$

3.3 Programmets flöde

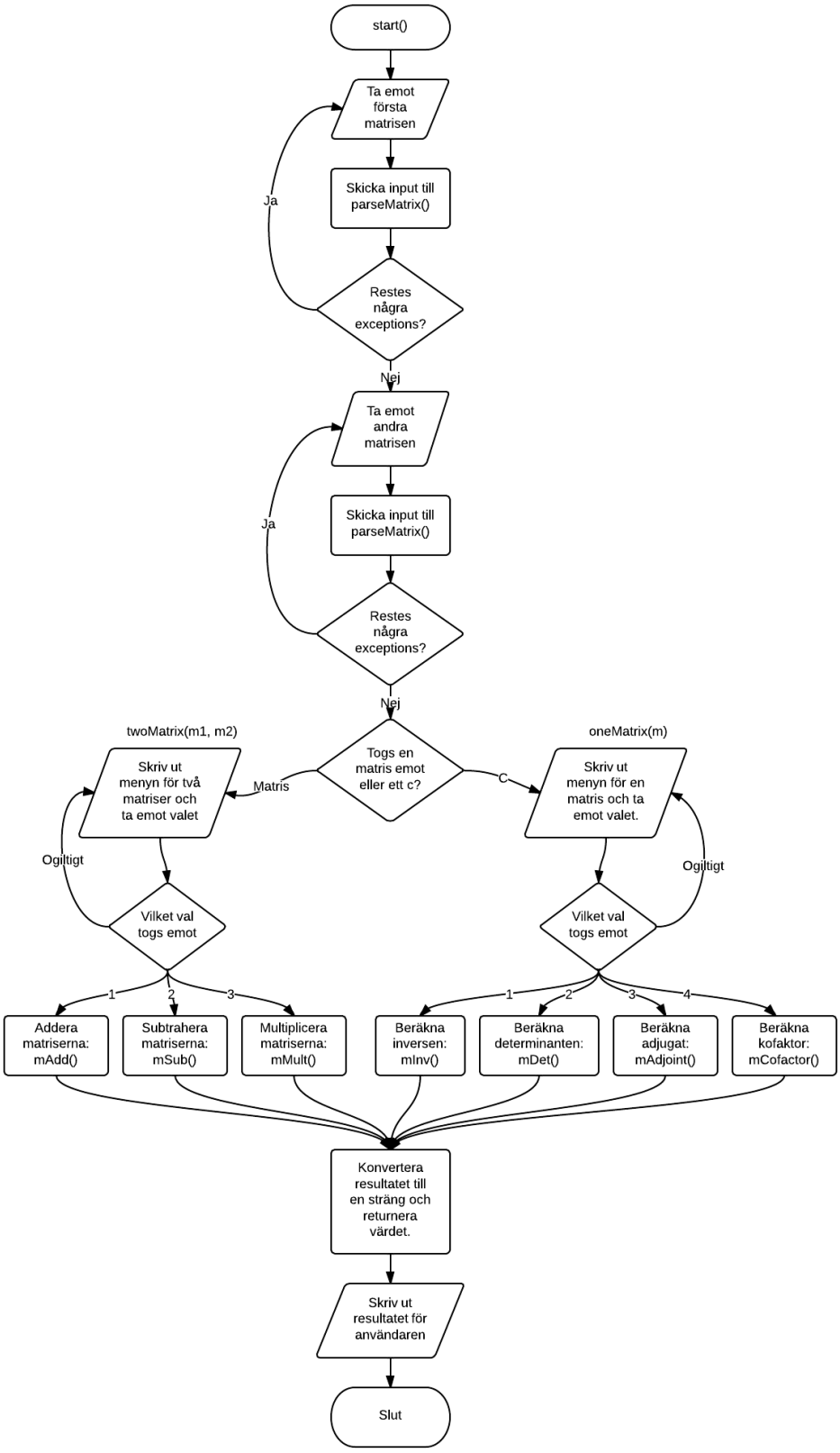
När användaren startar programmet genom att kalla på funktionen ”start()” i filen ”main.sml” börjar den med att be användaren om den första matrisen. När denna tagits emot skickas den vidare till funktionen ”parseMatrix(<indata>)” som försöker omvandla indatan till en matris. Skulle den misslyckas reser den ett felmeddelande, detta tas då emot utav start-funktionen och den ber användaren på nytt om att mata in en matris. När användaren matat in en matris som programmet kan läsa av utan att resa fel skickas matrisen vidare till funktionen ”secondMatrix(<matris>)” som ber användaren att mata in en andra matris. Här är det likadant som i start-funktionen med undantaget att användaren har möjlighet att mata in ett ”c” för att hoppa över inmatningen utav en andra matris.

Skulle det ske skickas den första matrisen vidare till funktionen ”oneMatrix(<matris>)” som ber användaren om att välja vilken uträkning denne vill ska utföras med matrisen. Användaren får valet mellan att beräkna inversen, determinanten, adjugatet, och kofaktormatrisen. Beroende på vad användaren väljer skickas matrisen vidare till den funktion som sköter den utvalda beräkningen och returnerar sedan värdet som omvandlas till en sträng. Strängen returneras tillbaka genom funktionskedjan och skrivs ut i terminalen utav start-funktionen. När detta skett avslutas programmet.

Skulle användaren mata in en andra matris skickas de båda vidare till funktionen ”twoMatrix(<matris 1>, <matris 2>)” som ber användaren välja vilken uträkning som ska ske med de två matriserna. Valet står mellan addition, subtraktion, och multiplikation mellan de två matriserna. När användaren gjort sitt val skickas de två matriserna vidare till respektive funktion. När returvärdet tagits emot konverteras det till en sträng och

returneras tillbaka genom funktionskedjan och skrivs till sist ut i terminalen utav start-funktionen och programmet avslutas.

För en mer visuell beskrivning, var god se flödesschema nedan.



Figur 1: Flödesschema

4 Slutsats

Programmet MatrixSolver är testat och ger ut rätt svar. För att veta att programmet ger ut rätt svar har resultatet jämförts med svar som hämtats via andra matrisberäknare på internet. Programmet arbetar snabbt för de vanligaste storlekarna på matriser. Vid ett test lät vi programmet hämta ut en $6 \cdot 6$ matris från en sträng och räkna ut determinanten för den 10 000 gånger, detta tog ungefär 3 sekunder. Testet tar givetvis olika tid beroende på vilken dator testet körs på, men det ger iallafall ett ungefärligt besked om hur effektivt programmet är.

4.1 Diskussion

När en användare ombeds att mata in en matris i programmet så måste varje tal matas in som bråk, även om de är heltal. Detta borde istället åtgärdas så heltal matas in som heltal och de tal som är bråk matas in som just bråk. Detta finns inte i den aktuella implementeringen utan kan tänkas som en vidareutveckling utav det.

En annan tänkbar vidareutveckling vore att snygga till hur resultatet skrivs ut i terminalen när programmet räknat klart. Just nu har den ett väldigt simpelt utseende och skulle mycket väl kunna snyggas till på något tänkbart sätt. Eventuellt skulle allt detta kunna ske via en hemsida, det vill säga, användaren matar in en matris i en hemsida, trycker på en knapp för att välja uträkning, och därefter skrivs resultatet ut på hemsidan snyggt och prydligt.

I aktuell utföring innehåller programmet endast relativt simpla matematiska algoritmer gällande matrisberäkning. Detta kan utökas enkelt då grunden för nya beräkningar är lagd. Det enda som behövs är tid för att implementera dem.