EFFEKTIV REPRESENTATION AV TIDSSERIER FÖR WEBBAPPLIKATIONER

Lars Sundman

Diplomarbete i datateknik
Handledare: Marina Waldén och Annamari Soini
Fakulteten för naturvetenskaper och teknik
Åbo Akademi
Juni, 2022

Sammanfattning

I diplomarbetet diskuteras några metoder för datareduktion av tidsserier i syfte att göra de ofta mycket långa serierna enklare att visualisera i webbläsare. Då en datavisualisering ska vara interaktiv utesluter det ofta att färdiga diagram lagras på webbservern, utan diagrammen ritas upp i webbläsaren medan användaren väntar. Utmaningen för att rita upp datavisualiseringar över stora mängder data i webbläsare ligger i att överföra data och att visualisera dem på ett sätt som tillåter interaktivitet men som ändå inte gör beräkningarna för resurskrävande. Vektorgrafik i SVG-format är en vanlig form för interaktiva diagram på webben eftersom de enkelt möjliggör att användaren på olika sätt manipulerar visualiseringen. För långa tidsserier blir SVG-diagrammen ändå för tröga och webbläsaren klarar inte av att animera dem utan att användaren hamnar vänta. Genom att lämna bort sådana punkter som är onödiga för visualiseringen kan en tidsserie representeras på ett tillförlitligt sätt. I arbetet presenteras fyra metoder för datareduktion och hur väl de fungerar jämförs. Både en visuell aspekt, hur väl de reducerade diagrammen efterliknar det ursprungliga, och en tidsmässig aspekt, helt enkelt hur lång tid diagrammen tar att producera, utforskas. Dessutom diskuteras metodernas lämplighet för tillämpning inom den sedvanliga arkitekturen för interaktiva webbapplikationer.

Sökord: webbsidor, datareduktion, tidsserier, datavisualisering

Innehållsförteckning

Förkortningar

DFT Discrete Fourier Transform 12–15, 32, 37–39, 42, 46–48, 51, 52, 56, 58, 60

DWT Discrete Wavelet Transform 15, 18, 32, 37–39, 43, 44, 46–48, 52, 53, 56, 58, 60

M4 datareduktion som i varje intervall väljer fyra extremvärden 20, 21, 32, 39, 42, 43, 46–51, 54, 56, 58–61, 67

MSE mean squared error 23, 24, 43, 44, 46–54

PAA Piecewise Aggregate Approximation 18, 19, 32, 38–40, 42, 43, 46–52, 54, 56, 58, 60, 61, 66

SSIM structural similarity index measure 23, 24, 39, 43, 44, 46–54, 56, 58, 59

SVG Scalable Vector Graphics 26, 30, 31, 33, 38–40, 43, 46–49

1 Inledning

Data och fakta är relaterade. De svenska pluralformerna kan härledas från latinets *datum* och *factum* som står för det som är *givet*, respektive det som genom en handling *uppstått* [1]. På ett vetenskapsfilosofiskt plan, om experiment antas fånga världen som den på riktigt är kan de data som experimenten producerar användas för att skapa fakta [2]. På det sättet används också data och datavisualisering i dagens samhälle. Data används för att fatta beslut, för att förstå omvärlden och, genom att övertyga, förändra den [3]. Datavisualiseringen som ett verktyg inom kommunikation har också fått en framstående roll i samhället, data samlas in för att visualiseras och förmedlas[4]. I takt med digitaliseringen har webben allt mera tagit över från dagstidningar och andra tryckta media som den viktigaste publiceringsplattformen för data och datavisualiseringar [5]–[7]. Detta betyder att webbutvecklare allt oftare är de personer som i dagens samhälle ansvarar för de tabeller och diagram som publiceras för allmänheten. Som webbutvecklare är det motiverat att studera datavisualisering för att förstå hur data bäst framställs i webbläsare.

Diagram och grafer har en förmåga att fästa läsarens uppmärksamhet och att effektivt förmedla information. Framställning av statistik i annan form än tabeller är ändå inte något som alltid varit självklart, tvärtom var skepsisen stor till en början. Datavisualiseringen har sitt ursprung i kartritning, från 1600-talet hittas det exempel på kartor som förutom geografin beskriver fenomen som passadvindarna och monsunområden, men redan mot slutet av 1700-talet utvecklades flera moderna diagram som linjediagram, stapeldiagram och kakdiagram för användning med ekonomiska data. Under början av 1800-talet exploderade intresset för datavisualisering och mot slutet av århundradet hade både statistiken som vetenskap och diagrammen erhållit en viktig roll för ekonomin och industrialiseringsprocessen i Europa. Under första hälften av 1900-talet gick intresset för diagrammen i vågor, med nya metoder för grafisk analys och datoriseringen av vetenskapen på 1970-talet fick de ändå ny luft under vingarna. I dagsläget är datavisualiseringen ett välutvecklat forskningsområde med användning inom många olika vetenskaper. [8, Kap. 1][9]

Interaktiva datavisualiseringar har också hittat till webbläsaren. Till webbsidor har det länge varit möjligt att bifoga bildmaterial vid sidan om textinnehållet, t.ex. har bildelementet `img` funnits i webbläsare sedan Mosaic kom ut år 1993, och stödet för SVG-illustrationer har varit utbrett sedan 2005 [10]. I takt med att webbläsarnas programmeringsomgivningar standardiserats och blivit mer välutvecklade har också webbsidor blivit mer och mer interaktiva. I dagens läge utvecklas det regelbundet webbapplikationer som i sina funktioner t.o.m. kan ersätta traditionella programpaket för persondatorer [11]. Webbsidor har dessutom förblivit lätta att förmedla, en användare behöver bara skickas en HTTP-länk för att sedan, inom några sekunder, ha möjligheten att studera materialet. I det här sammanhanget är det inte konstigt att fler och fler datavisualiseringar, och i synnerhet

interaktiva sådana, förmedlas på webben istället för i tryck, att skicka en datavisualisering över webben kräver inte ens att skaparen har en egen webbserver. Webbsidor som Observable låter användaren skapa och dela interaktiva datavisualiseringar utvecklade i JavaScript, och vetenskapliga verktyg som Python-paketet Jupyter och R-verktyget RStudio låter forskaren arbeta med s.k. *notebooks* (sv. anteckningsblock) där både resultat och process kan presenteras i ett dokument och delas på webben.

Det finns ändå utmaningar med att visualisera data i webbläsaren. Om data inte kan förbehandlas på servern utan både datavisualisering och en automatisk dataanalys sker i slutanvändarens webbläsare finns det en risk att intrycket av visualiseringen inte blir det önskade. Om t.ex. användarens internetuppkoppling är långsam kan överföringen av data ta för lång tid, eller om användarens dator eller smarttelefon inte klarar av att utföra de nödvändiga beräkningarna tillräckligt snabbt finns det en risk för att interaktiviteten haltar. För en framgångsrik datavisualisering ska läsarkretsen tas i beaktande, t.ex. är vågräta diagram med vågrät text lättare att läsa än diagram med lodrät text, färger ska vara lämpliga också för färgblinda läsare och förkortningar ska förklaras [8, s. 183–190]. För en interaktiv datavisualisering på webben är det också viktigt att ta i beaktande läsaren, men utöver de rent utseendemässiga kraven som gäller för alla datavisualiseringar oberoende om de är i tryck eller digitala kräver en datavisualisering på webben att slutanvändarens datorkraft, skärmupplösning och internetuppkoppling också beaktas.

För stora samlingar data är det ofta enklast att bara överföra en del data till användarens webbläsare, i praktiken ligger det nära till hands att bara överföra en analys till användarens webbläsare. När det t.ex. kommer till att rita upp ett klimatdiagram över Åbo ska all statistik över lufttemperatur och nederbörd genom åren sammanfattas till ett diagram. Då kan det kännas mer naturligt att bara överföra den färdiga analysen, dvs. medeltemperaturen och -nederbörden för varje månad, till webbläsaren. Detta är ändå ingen allmän lösning. Om målet istället för klimatdiagrammet är att visa ett linjediagram över temperaturutvecklingen i Åbo genom åren, utan någon sammanfattning över månader eller år, skulle en analys *kringgå* utmaningen och istället ge en illustration som avviker från den ursprungliga uppgiften. Den här problematiken föranleder en av frågorna i detta arbete; är det möjligt att hitta en allmän lösning för att tillräckligt snabbt överföra de data som krävs för en datavisualisering?

Då bilder används i webbsidor används ofta formatet PNG eller JPEG. En intressant egenskap för båda formaten är att de utgör komprimerade bilder, dvs. när bilder i dessa format används i webbläsaren sker först en avkodningsprocess som packar upp dem innan de kan visas. Formatet PNG använder sig av komprimeringsalgoritmen Deflate som inte leder till tappad information när bilder komprimeras, processen är t.ex. jämförbar med hur text kan komprimeras med algoritmer som LZ77 där upprepade förekomster av

tecken ersätts med hänvisningar till en enda plats i data där det tecknet lagrats. Formatet JPEG utvecklades i sin tur kring en komprimeringsalgoritm som bygger på den diskreta cosinustransformen och strävar till att tappa sådan information som är onödig då bilden visas för en människa. En intressant detalj att känna till är att alla textdata som skickas från en webbserver till en webbläsare oftast också är komprimerade med Deflate eller en annan komprimeringsalgoritm som inte tappar information, datorer har blivit så snabba på att komprimera och att packa upp data att den här typen av komprimering helt enkelt anses höra till när data skickas över Internet. Data som används för interaktiv datavisualisering på webben skickas ofta också som text, t.ex. som JSON-filer, och är därför nästan alltid komprimerade i det skede de lämnar webbservern. I praktiken är detta ändå inte tillräckligt, för stora mängder data blir överföringstiden för lång. Dessa fakta föranleder nästa fråga, i samma stil som JPEG ger mer komprimerade bilder än PNG, finns det någon metod för att komprimera data som skulle ge ett bättre resultat genom att tappa för en visualisering irrelevanta data?

En intressant kategori av data som skulle kunna användas för en undersökning inom dessa ramar är tidsserier. Vanliga tidsserier som påträffas på nätet består ofta av stora mängder data, t.ex. serier som beskriver utvecklingen för en aktiekurs eller temperaturmätningar över tid. Tidsserier är relevanta inom vetenskapen men de är också intressanta för allmänheten. Som linjediagram kan de också visualiseras på ett sätt som synliggör trender och extremvärden också för de som inte är insatta i ämnet.

I detta arbete undersöks några metoder för datareduktion. Målet för arbetet har varit att hitta metoder som låter en webbutvecklare komprimera data på ett sätt som inte avsevärt förändrar det resulterande linjediagrammet som ritas upp i en interaktiv datavisualisering i en slutanvändares webbläsare. Genom att på detta sätt komprimera data kan nämligen en webbutvecklare, som kanske inte är expert på de fenomen som visualiseras, skapa en datavisualisering över data utan att mer än nödvändigt ta ställning till hur datan borde sammanfattas eller analyseras. Metoderna tillämpas på tidsserier och resultatet jämförs med bilder på referensdiagram med hjälp av mått som mäter likhet mellan bilder. Hur enkla metoderna är att tillämpa i den omgivningen som är vanlig för en webbsida eller -applikation diskuteras också.

2 Ändamålsenliga diagram

För att dra slutsatser av, och kommunicera egenskaper för, data används olika statistiska metoder, både matematiska och grafiska. Till de matematiska metoderna hör t.ex. beskrivande statistik, där lägesmått och spridningsmått sammanfattar insamlade data, och inferentiell statistik, där stickprov och fördelningar används för att uppskatta egenskaper för ännu okända data. [12]

De grafiska metoderna för att kommunicera statistiskt material går under det allmänna begreppet datavisualisering, med vilket avses alla de metoder som framställer data i bilder istället för tabeller. Datavisualisering används ofta för att kommunicera statistiskt material, men förutsatt att en visualisering är ändamålsenlig kan den också användas för att dra särskilda slutsatser ur data. Metoderna som utnyttjar diagram och grafer för att på ett vetenskapligt sätt utforska data brukar falla under begreppet explorativ dataanalys [eng. *exploratory data analysis*]. Dessa metoder kännetecknas av att de tar avstamp i den beskrivande statistiken och strävar till att använda grafiska verktyg för att ställa upp modeller. [13]

För att kommunicera data är det viktigt att data presenteras på ett effektivt och sanningsenligt sätt. För dataanalys är det speciellt viktigt att data presenteras på ett sätt som inte förvränger de underliggande resultaten.

I boken *The Visual Display of Quantitative Information* diskuterar Edward Tufte vad som kännetecknar s.k. ”utomordentlig grafik” (*Graphical Excellence*) [8, s. 13]. Den ursprungliga upplagan av boken, och det mesta av innehållet, skrevs i ett sammanhang där diagram uppritade av datorer ännu var relativt enkla, men boken innehåller ändå råd som är relevanta också för grafer som ritas upp med hjälp av moderna datorer och i webbläsaren.

Tufte skriver att diagram som fyller sin funktion som ett effektivt verktyg för kommunikation kännetecknas av att de framställer intressanta data med eftertanke, genomgående analys och en slående design. Diagram ska effektivt kommunicera komplicerade idéer med klarhet och precision. De ska förmedla så många idéer som möjligt på en så kort tid som möjligt.

Sanningsenliga diagram kännetecknas enligt Tufte av att data sätts i rätt sammanhang och med eftertanke gällande proportionerna. Den grafiska representationen av ett tal ska ta plats i rätt förhållande till talets storlek i jämförelse med resten av data. Representationen ska inte heller ta upp plats i fler dimensioner än de variabler som existerar i data, dvs. om t.ex. ett stapeldiagram uttrycker en viss variabel längs med y-axeln i diagrammet ska inte denna samma variabel påverka staplarnas bredd längs med x-axeln. Om ingen skild variabel används för att variera bredden på staplarna ska alla staplar också följaktligen ha samma bredd så att inte läsaren föranledes att tro att bredden på staplarna har någon be-

tydelse i datan. [8, s. 77]

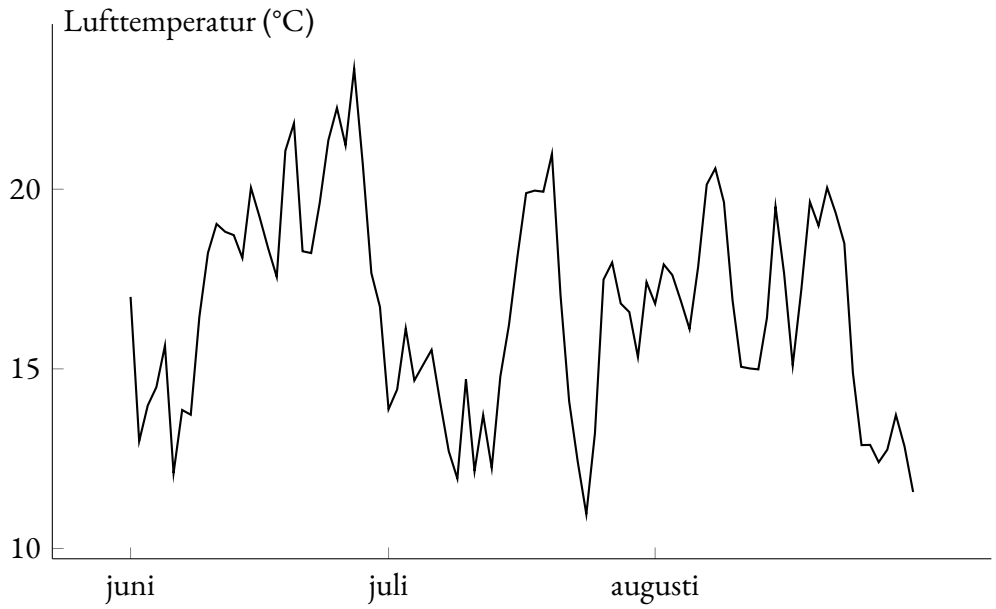
$$\text{Lögnfaktor} = \frac{\text{effektens utsträckning i grafiken}}{\text{effektens storlek i data}} \tag{1}$$

För att sätta en siffra på hur mycket ett diagram förvränger data presenterar Tufte termen lögnfaktor, här översatt till svenska i ekvation 1. Lögnfaktorn uttrycker hur mycket ett diagram ”ljuger” genom att mätpunkterna inte är proportionerligt uppritade. [8, s. 57]

Effektens utsträckning i visualiseringen ska mätas helt praktiskt, med t.ex. en linjal på papper, medan effektens magnitud i data kräver att mätpunkter jämförs med varandra. Enligt Tufte gäller det att bildmaterial som över- eller underskattar data med 5 %, det vill säga med en lögnfaktor $LF > 1{,}05$ respektive $LF < 0{,}95$, avsevärt förvränger data och gör illustrationen oanvändbar. Ett diagram kan alltså ritas upp i olika storlekar men det är viktigt att skalorna på axlarna och förhållandet mellan mätpunkternas storlekar också beaktas då diagrammet förstoras eller förminskas.

För att effektivt kommunicera statistisk information är det viktigt att sätta sig in vad mottagaren förväntar sig läsa och höra. Då är det också viktigt att de diagramtyper som används är tillräckligt bekanta för mottagaren och att de används på rätt sätt. Här följer beskrivningar av några vanliga diagramtyper.

2.1 Linjediagram



Figur 1: Medeltemperaturen vid Åbo flygplats under sommaren 2020

Ett linjediagram beskriver utvecklingen i en mätvariabel y över tid, eller på ett mera allmänt plan över en sekvens av på varandra följande mätningar. Linjen mellan punkterna fungerar dels som en uppskattning av värdet i y mellan mätningar, och dels som ett sätt

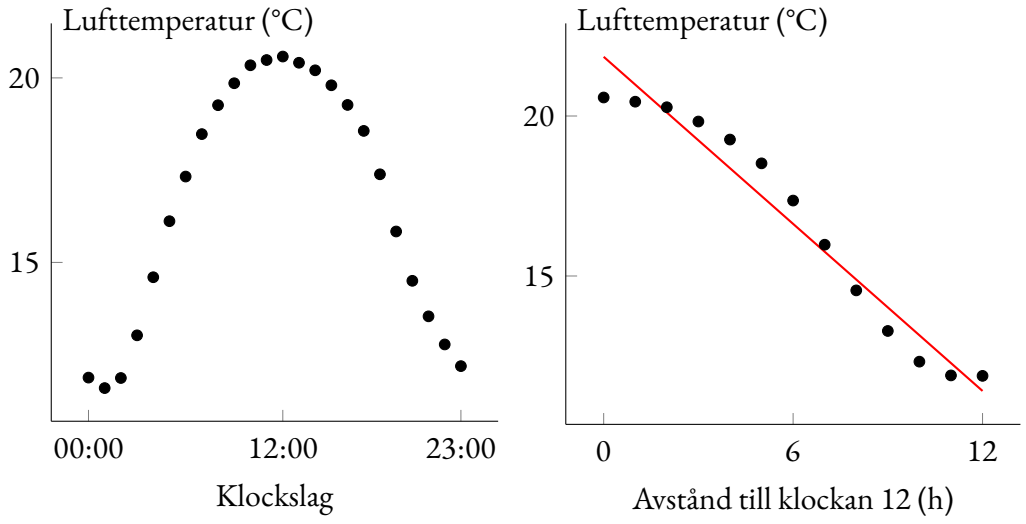
att lyfta fram förändring och trender i data. I det här hänseendet avviker linjediagrammet från punktdiagram, för vilka uppgiften oftast är att visa sambandet mellan variablerna i ett visst ögonblick, även då det också finns mera sällsynta former av punktdiagram där förändring visas med linjer. [14, s. 184–185]

Linjediagram är speciellt användbara för att åskådliggöra fenomen över en längre tid. Med ett stort antal mätpunkter blir diagrammet lätt oöversiktligt om alla punkter ritas ut med cirklar eller kryss som i ett punktdiagram. I ett överskådligt linjediagram över stora mängder data lämnas också med fördel markörerna för punkterna bort. Den enkla linjen som återstår uppmuntrar läsaren att sätta de enskilda mätpunkterna till sidan och istället se utvecklingen i data. [14, s. 184–185]

Figur 1 visar medeltemperaturens utveckling på dygnsbasis vid flygplatsen i Åbo under sommaren 2020. Ur diagrammet går det inte att precis avgöra värdet i något visst ögonblick, men större fenomen som t.ex. de varmare perioderna i slutet på juni och i mitten och slutet på juli går att urskilja.

2.2 Punktdiagram

Ett punktdiagram beskriver sambandet mellan två variabler, x och y, där varje punkt är en relation (x_i, y_i) i en serie mätningar. Det utmärkande för punktdiagrammet är att x-axeln inte visar ordningen i vilken punkterna blivit uppmätta, utan båda variablerna kan variera fritt. [12]



(a) Korrelation (b) Kausalitet

Figur 2: Temperaturen som en funktion av tiden på dygnet

Punktdiagram är vanliga inom naturvetenskaperna och används ofta för att analysera orsak och verkan [8, s. 85][15, kap. 8]. I en studie av orsakssamband, dvs. *kausalitet*, kallas i regel den variabel som studeras för mätvariabel eller responsvariabel och sätts till y, me-

dan den kända variabeln, som under experimentets gång kontrolleras, kallas för kontrollvariabel eller förklarande variabel och sätts till x. Ett viktigt steg i att evaluera ett möjligt orsakssamband i ett punktdiagram är att i samma diagram rita upp en på förhand utarbetad matematisk modell över sambandet och sedan jämföra hur nära punkterna följer modellen [8, s. 49][13, kap. 5].

Om kausaliteten är okänd, t.ex. om det saknas en modell för att förklara variablernas förhållande till varandra, eller ett orsakssamband inte är målet för studien, kan både y och x användas som mätvariabler. I detta fall kallas variablernas samband för *korrelation* och ofta används en linje genom punktdiagrammets diagonal utgående från origo som en modell för fullständig korrelation. Avståndet mellan punkterna och diagonalen kan beskriva hur variablernas variation förhåller sig till varandra. [14, s. 190–191][15, s. 249–256]

Figurerna 2 (a) och 2 (b) visar hur punktdiagram kan användas för att studera hur lufttemperaturen varierar under dygnet. I figur 2 (a) har varje dygn under hela sommaren 2020 delats in i 24 timmar och medeltemperaturen för dessa beräknats. Genom att jämföra klockslaget med medeltemperaturen fås de 24 punkterna som är uppritade i figur 2 (a). Ur figuren går det att utläsa att det är som kallast ungefär klockan ett eller två under morgonnatten och som varmast kring klockan tolv och klockan ett mitt på dagen.

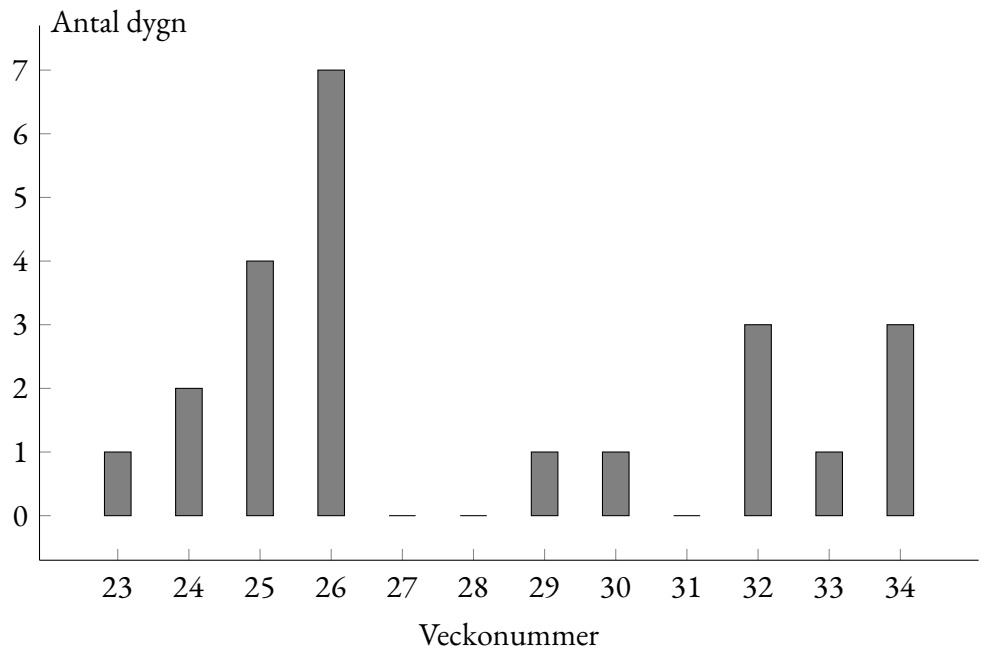
I praktiken finns det förstås många faktorer som påverkar hur varmt det är på någon viss plats, men en förenklad modell som kunde förklara fenomenet i figur 2 (a) är att medeltemperaturen är omvänt proportionell mot avståndet till klockan tolv i ett visst klockslag, dvs. temperaturen blir lägre ju högre avståndet till klockan tolv är.

I figur 2 (b) har mätningarna under sommaren delats in i klasser enligt hur långt de i jämna timmar befunnit sig från klockan tolv på dagen. Sedan har medelvärdet för varje klass beräknats. Ur diagrammet går det att utläsa att temperaturen faktiskt sjunker i takt med att avståndet till klockan tolv ökar. Regressionslinjen verkar också passa mätningarna väl, vilket tyder på att dett uppmätta fenomenet kanske motsvarar modellen.

2.3 Stapeldiagram

Två olika varianter av stapeldiagram är vanliga, det lodräta diagrammet där staplarna växer från x-axeln upp längs med y-axeln och det vågräta diagrammet där staplarna utgår ifrån y-axeln och växer längs med x-axeln. Ett lodrätt stapeldiagram passar data där mätvariablerna x och y båda två är kvantitativa variabler, medan ett vågrätt stapeldiagram passar data där x är kvantitativ och y är kvalitativ. [14, s. 180, 182]

I allmänhet används stapeldiagram för att framställa kategoridata, det vill säga data där den förklarande variabeln är diskret och beskriver något gemensamt för många datapunkter [14, s. 41]. Analysen som utförs i stapeldiagram är följaktligen ofta sammanfattande.



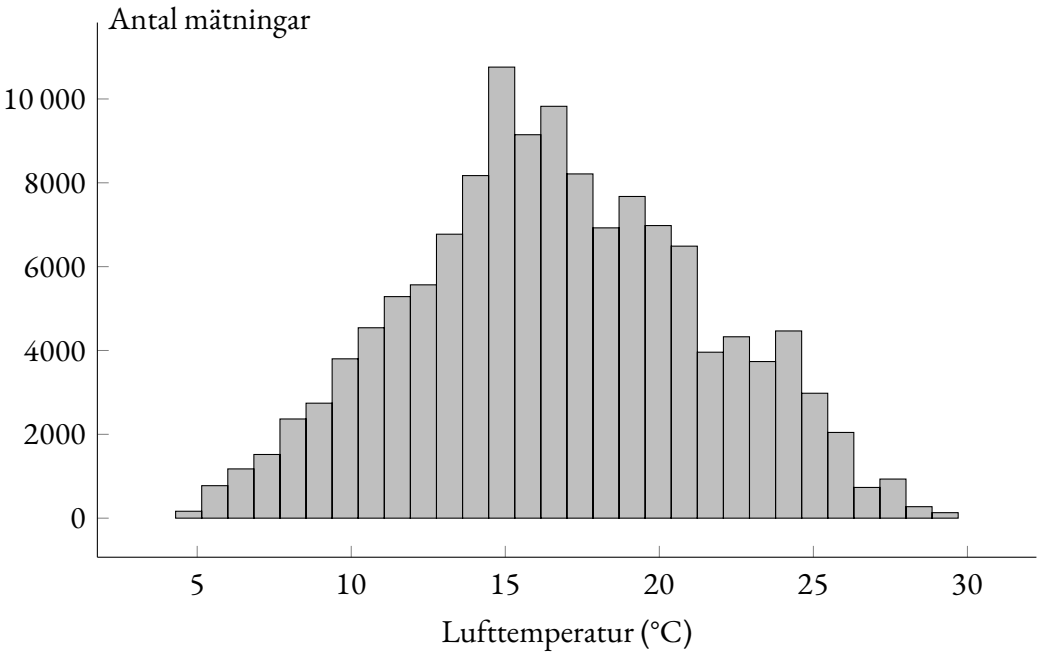
Figur 3: Dygn med värmebölja under sommarens veckor

Figur 3 är ett exempel på ett lodrätt stapeldiagram. Figuren är en sammanfattning som visar vilka veckor under sommaren som haft dygn då en lufttemperatur högre än 25 °C uppmätts, dvs. dygn med värmebölja. Enligt figuren verkar juli (v. 28–31) ha haft färre sådana dygn än både juni (v. 23–27) och augusti.

2.4 Histogram

Histogram är lodräta stapeldiagram som beskriver hur en variabel är distribuerad. För att rita upp ett histogram delas data först in i klasser enligt mätresultaten för någon variabel, varefter antalet observationer inom varje klass noteras. Därefter ritas histogrammet upp som ett stapeldiagram med variabeln x som antar värden ur mängden klasser och variabeln y som antalet observationer inom de enskilda klasserna. Inom sannolikhetsläran är histogram användbara eftersom de på ett effektivt sätt visuellt kan uppskatta fördelningsfunktionen för en stokastisk variabel. Så länge varje stapel har likadan bredd och skalan på y-axeln också är jämn kommer ytan som histogrammet upptar approximera ytan under fördelningsfunktionens kurva. Ett histogram hjälper läsaren få en uppfattning om hur sannolika olika utfall är. [14, s. 182][12]

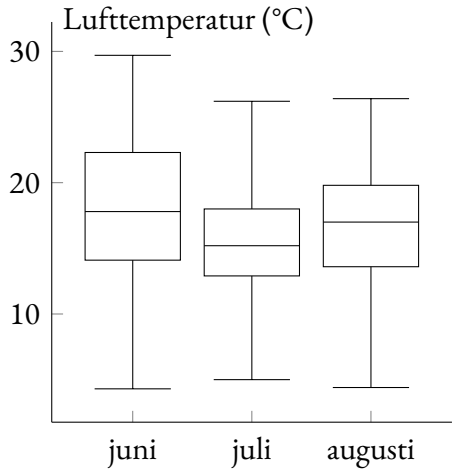
Figur 4 är ett histogram över temperaturmätningarna under sommaren. Temperaturmätningarna har delats in i 30 klasser, vilket betyder ungefär en klass per grad på skalan, sedan har antalet mätningar i varje klass beräknats. Ur figuren kan utläsas att klassen med flest medlemmar är strax under 15 °C och diagrammets tyngdpunkt verkar ligga nära den klassen. Medeltemperaturen och mediantemperaturen verkar alltså utgående från histogrammet ligga nära varandra, med resten av mätningarna ungefär jämnt fördelade på båda



Figur 4: Fördelning för sommarens temperaturmätningar

sidor om medelvärdet. Ur figur 4 framgår alltså att temperaturmätningarna möjligen följer en normalfördelning.

2.5 Lådagram



Figur 5: Beskrivande statistik över sommarens temperatur

Ett lådagram (eng. *box plot*), är på samma sätt som histogrammet ett diagram som beskriver fördelningen för en variabel i data. Den grundläggande modellen är ett diagram där varje variabel illustreras som en rektangel, *lådan*, med två streck, *morrhår*, som utgår ifrån de övre och nedre kanterna på lådan. Lådagrammet beskriver variabeln med hjälp av fem tal: medianen, första kvartilen (dvs. 25 % av mätningarna), tredje kvartilen (75 %),

minimum och maximum. Lådan har sina kortsidor i de två kvartilerna, medianen ritas upp som ett streck tvärs över lådan och minimum och maximum representeras av morrhårens utsträckning från kvartilerna. Uteliggare kan ritas in som bollar eller streck utanför lådan. [15, s. 197]

Figur 5 beskriver hur temperaturmätningarna fördelats under juni, juli och augusti. Ur diagrammet kan utläsas att mediantemperaturen under juni månad var strax under 20 °C medan den under juli var lägre, ungefär 15 °C, och i augusti lite under mediantemperaturen i juni. Extremvärdena verkar ha varit ungefär de samma under juli och augusti, med den lägsta temperaturen kring 5 °C och den högsta kring 26 °C. I juni verkar temperaturen ha varit mer utspridd än i juli. I båda fallen låg den första kvartilen kring 15 %, men temperaturen i juli i 75 % av mätningarna var under ca. 19 °C medan gränsen för samma kvartil i juni var kring 23 °C.

3 Tidsserier och datareduktion

En tidsserie består av flera på varandra följande observationer, i tidsordning eller helt enkelt i den ordning de samlats in. En tidsserie T är alltså en följd av n antal variabler som antar reella värden, och kan sammanfattas som i följande ekvation. [16]

$$T = (t_1, \ldots, t_n)\ t_i \in \mathscr{R} \tag{2}$$

Tidsserier förekommer inom flera discipliner. Inom statistiken och sannolikhetsläran används tidsserier för att beskriva stokastiska processer som till exempel en valutakurs eller temperaturförändringar. Till den statistiska tidsserieanalysen hör metoder och modeller för att bland annat karaktärisera tidsserier och deras underliggande fenomen samt för att förutspå deras framtida utveckling. [17]

Inom tekniken är begreppet signal nära besläktat med tidsserier. Signaler överför information, de representeras med liknande matematik som tidsserier och problemet att skilja på brus och signal är ur en viss synvinkel samma problem som att för en tidsserie definiera en underliggande modell. Till skillnad från tidsserieanalysen ligger fokuset i signalbehandlingen ändå på hanteringen av konstruerade och deterministiska signaler. [17]

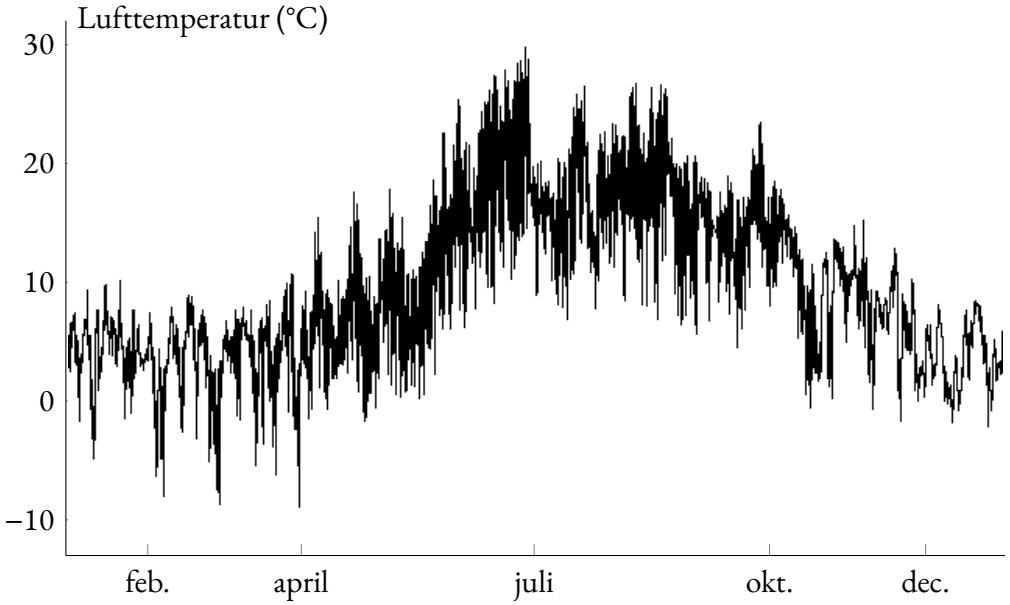
Metoder för datareduktion är viktiga inom datautvinning (eng. *data mining*) och i synnerhet då målet är att utvinna data ur tidsserier. Målet med datareduktion är att representera data i en mer kompakt form och på så sätt tillåta användningen av sådana analysmetoder och algoritmer som annars skulle vara för resurskrävande i sina beräkningar. Ur ett praktiskt databasperspektiv handlar det om att minska på antalet rader eller kolumner i data. [18, s. 37]

Termen *dimension* är vanligt förekommande inom dataurvinning. Om varje tupel (rad) i en relationsdatabas är en del av en relation (tabell) som beskrivs av n-antal attribut (kolumner) sägs tupeln ha n-antal dimensioner [18, s. 3]. En vanlig form av datareduktion inom datautvinning är *dimensionalitetsreduktion*. Insikten bakom dimensionalitetsreduktion är att verkliga data ofta har redundanta dimensioner. Målet är att lämna bort de dimensioner vars information kan härledas från andra kolumner, och på det sättet reducera antalet dimensioner till endast de som är nödvändiga för att informationen i data kan förmedlas. Ett enkelt exempel är en kolumn för en persons ålder i en databas, den kan reduceras bort genom att istället uttrycka samma information med hjälp av en kolumn som sparar personens födelsedatum om en sådan också hittas i databasen. [18, s. 41]

För tidsserier är begreppet dimension ändå inte helt entydigt, meningen beror på användningsområdet [18, s. 459]. Inom dataurvinningen kategoriseras tidsserier som antingen *univariata*, med endast ett objekt eller system som observeras, eller *multivariata*, med flera. I multivariata tidsserier betraktas dessa olika undersökningsobjekt som dimen-

sioner på samma sätt som i andra data, och en dimensionalitetsreduktion följer samma mönster. I univariata tidsserier brukar de enskilda tidpunkterna betraktas som dimensioner, och dimensionalitetsreduktion sker på ett allmänt plan genom att uttrycka mätpunkterna med hjälp av antingen de föregående eller de påföljande punkterna i serien. [18, s. 462]

Målet med en datarepresentation är att åstadkomma dimensionalitetsreduktion jämfört med den ursprungliga tidsserien och på så sätt spara beräkningsresurser. I allmänhet ska en datarepresentation åstadkomma en signifikant reduktion i dimensionalitet, ha en betoning på den fundamentala formen på data på både lokal och global nivå, ha en låg beräkningskostnad, kunna rekonstruktrueras med en ändamålsenlig upplösning och inte vara känslig för brus. [16]



Figur 6: Temperaturen vid Åbo flygplats under år 2020

Målet med de följande avsnitten är att beskriva fyra intressanta metoder för datareduktion. För att illustrera de olika metoderna används här en tidsserie med ungefär 520 000 temperaturmätningar uppmätta vid Åbo flygplats under tiden 1.1.2020 till 31.12.2020. Tidsserien är hämtad via Meteorologiska institutets dataportal ([19]) och är uppritad i sin helhet i figur 6 för att kunna jämföras med exemplen i de avsnitt som följer.

3.1 DFT

I tidsplanet beskrivs en digital signal av dess amplitud vid diskreta samplingstidpunkter. För vissa tillämpningar är signalens frekvensinnehåll också intressant. Den diskreta Fouriertransformen [eng. *Discrete Fourier Transform* (DFT)] är en algoritm som omvandlar serien från sampel i tidsplanet till signalkomponenter i frekvensplanet. Den har visat sig an-

vändbar inom många forskningsområden, bl.a. telekommunikation, och ljud- och bildbehandling. [20, kap. 4]

På samma sätt som den kontinuerliga Fouriertransformen grundar sig det diskreta fallet på att en signal kan uttryckas med hjälp av periodiska sinusfunktioner. Härledningen av både den kontinuerliga och den diskreta Fouriertransformen kräver en längre diskussion, men eftersom detta arbete inte tillämpar transformen för frekvensanalys annat än för att välja rätt koefficienter för datareduktion räcker en uppfattning av algoritmen på en allmän nivå.

DFT av en signal x_n med längden n resulterar i en lika lång följd X_k och definieras som

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{i2\pi}{N}kn}. \tag{3}$$

Med följden X_k kan den ursprungliga signalen återskapas som

$$x_n = \frac{1}{N} \sum_{n=0}^{N-1} X_k e^{\frac{i2\pi}{N}kn}. \tag{4}$$

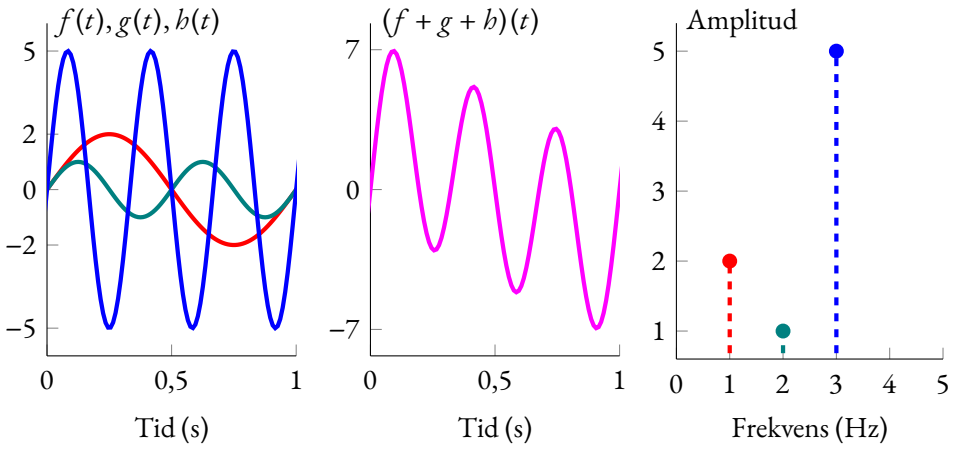
Målet med att tillämpa DFT är som sagt att undersöka signalen i frekvensplanet. Talen i följden X_k kallas för koefficienter, de motsvarar koefficienterna framför termerna i Fourierserien, och varje koefficient beskriver en signalkomponent, dvs. en sinusvåg, som funktion av en viss frekvens. Koefficienterna är komplexa vektorer och innehåller information om fas, som vinkeln mellan det reella och det komplexa talplanet, och amplitud, som vektorns magnitud. Amplituden är relativ till de andra amplituderna i signalen. [20, kap. 4]

Som ett exempel, funktionerna

$$\begin{aligned} f(t) &= 2sin(2\pi t) \\ g(t) &= sin(2\pi 2t) \\ h(t) &= 5sin(2\pi 3t) \\ \left(f+g+h\right)(t) &= 2sin(2\pi t) + sin(2\pi 2t) + 5sin(2\pi 3t). \end{aligned}$$

beskriver tre sinusvågor med olika amplituder och frekvenser samt en signal som är en kombination av dem. Definitionerna kan jämföras med det allmänna uttrycket för en harmonisk oscillator $y(t) = Asin(2\pi ft + \varphi)$ där A är amplituden, f frekvensen och φ fasförskjutningen.

För att rita upp funktionerna i figur 7 (a) har signalerna samplats med en samplingsfrekvens på 400 Hz. Det andra diagrammet i figur 7 (b) visar summan av de tre signalerna. Det tredje diagrammet i figur 7 (c) visar vilka frekvenser och amplituder som summasignalen består av. Frekvenskomponenterna 0, 4 och 5 Hz för vilka amplituden är 0 är ute-



(a) Tre signaler (b) Summasignal (c) Komponenter i summan

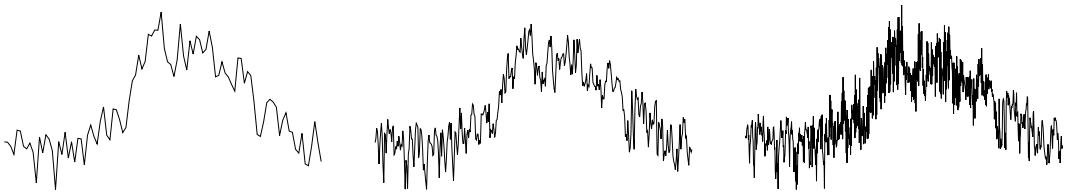
Figur 7: Frekvensanalys

lämnade ur diagrammet.

I följande uträkningar är koefficienterna avrundade till två gällande siffror för att spara utrymme. Eftersom den kombinerade signalen representeras med 400 sampel består också följden X_k efter DFT av 400 koefficienter. I koefficienternas magnituder är som sagt amplituderna för signalkomponenterna kodade. Genom att för varje tal i följden koefficienter först beräkna absolutbeloppet för det komplexa talet och sedan skala värdet med faktorn 2/400, som kommer av samplingsfrekvensen, erhålls de ursprungliga amplituderna för varje frekvenskomponent. Koefficienterna med index 2, 3 och 4 motsvarar frekvenserna 1,2, och 3, i detta fall med enheten Hertz. Uträkningen sker som följande. För att spara utrymme har de andra frekvenskomponenterna, som alltså i detta fall har en amplitud lika med noll, lämnats bort.

$$
\begin{aligned}
X_k &= \{\ldots \quad 3{,}2 - 4 \times 10^2 i \quad 3{,}2 - 2 \times 10^2 i \quad 2{,}3 \times 10^1 - 10 \times 10^2 i \quad \ldots\} \quad | \text{ Abs} \\
X_{Abs} &= \{\ldots \quad 4 \times 10^2 \quad 2 \times 10^2 \quad 10 \times 10^2 \quad \ldots\} \quad | \times 2/400 \\
X_{Amp} &= \{\ldots \quad 2 \quad 1 \quad 5 \quad \ldots\}
\end{aligned}
$$

Detta exempel illustreras i figur 7.



(a) 100 koefficienter (b) 400 koefficienter (c) 800 koefficienter

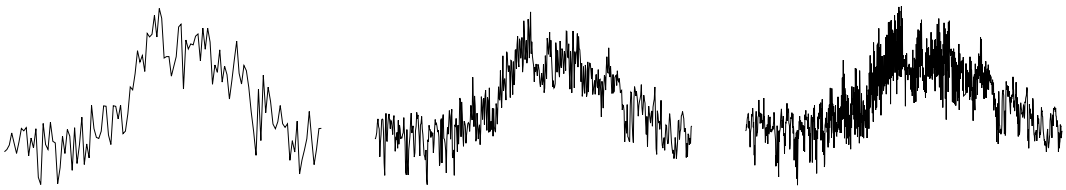
Figur 8: Figur 6 representerad med DFT i några upplösningsnivåer

I följden X_k utgör de tidigare koefficienterna lägre frekvenser och de senare högre. För

många naturliga signaler som t.ex. bilder har det visat sig att de lägre frekvenserna innehåller intressant information om överhängande trender och större strukturer medan de högre frekvenserna främst innehåller detaljinformation [21]. Det här motiverar DFT som ett verktyg för datareduktion. Genom att först beräkna en DFT för en tidsserie i sin helhet kan sedan serien förenklas genom att beräkna inversen men endast med en önskad andel av följden koefficienter. Längden på den återskapade tidsserien är lika med längden på följden av koefficienter som används för att beräkna den inverterade transformen. Figur 8 illustrerar den här metoden med hjälp av temperaturserien. Med färre koefficienter blir trenden för tidsserien klar, med fler tillkommer det mera detaljer, speciellt i partierna kring sommaren där dygnsvariationen verkar ha varit lägre.

Denna metod behandlas vidare i detta arbete. För enkelhetens skull används förkortningen DFT inte bara för transformen utan också för att omfatta metoden för datareduktion som utnyttjar den diskreta Fouriertransformen.

3.2 DWT



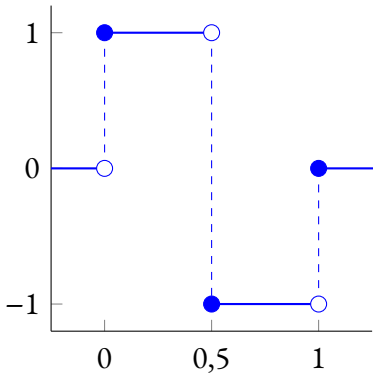
(a) Nivå 12 – 132 punkter (b) Nivå 10 – 514 punkter (c) Nivå 9 – 1020 punkter

Figur 9: Figur 6 representerad med DWT i några upplösningsnivåer

Den diskreta krusningstransformen [eng. *Discrete Wavelet Transform* (DWT)] är en transform som bygger på basfunktioner som kallas för krusningar (eng. *wavelets*). Krusningar är matematiska funktioner som representerar data som additioner och subtraktioner från en funktionsprototyp, en s.k. *moderkrusning*. Algoritmer som är baserade på krusningar kan hantera serier med olika upplösningar i olika partier eftersom varje krusning är tidslokal och kan representera ett intervall av valfri längd. [22], [23]

I Chan och Fu [22] presenteras en metod för datarepresentation som använder sig av DWT med den s.k. Haarkrusningen. Metoden använder sig av ett modifierat geometriskt avstånd med förskjutning i värde och tid som likhetsmått[24]. I Chan och Fu [22] presenteras ett bevis för att det geometriska avståndet mellan två tidsserier är det samma efter DWT, vilket är ett viktigt resultat för en användning av transformen för dataurvinning.

Den diskreta krusningstransformen kan tillämpas med flera olika krusningar, men Haarkrusningen används ofta som ett exempel. Haarkrusningens moderkrusning $\psi(t)$ kan de-



Figur 10: Haarkrusningen

finieras som

$$\psi(t) = \begin{cases} 1 & 0 \leq t < \frac{1}{2} \\ -1 & \frac{1}{2} \leq t < 1 \\ 0 & \text{annars} \end{cases} \tag{5}$$

och illustreras i figur 10. Skalningsfunktionen $\varphi(t)$ kan definieras som

$$\varphi(t) = \begin{cases} 1 & 0 \leq t < 1 \\ 0 & \text{annars} \end{cases} . \tag{6}$$

För alla heltalspar i,j kan en funktion definieras som

$$\psi_{i,j}(x) = \psi(2^j x - i). \tag{7}$$

För att illustrera Haartransformen, ta t.ex. ekvationen

$$f(t) = \begin{cases} 1 & 0 \leq t < \frac{1}{8} \\ 2 & \frac{1}{8} \leq t < \frac{2}{8} \\ 3 & \frac{2}{8} \leq t < \frac{3}{8} \\ 4 & \frac{3}{8} \leq t < \frac{4}{8} \\ 5 & \frac{4}{8} \leq t < \frac{5}{8} \\ 6 & \frac{5}{8} \leq t < \frac{6}{8} \\ 7 & \frac{6}{8} \leq t < \frac{7}{8} \\ 8 & \frac{7}{8} \leq t < \frac{8}{8} \end{cases}$$

som beskriver en diskret funktion $f(t)$ i intervallet [0,1]. Funktionen $f(t)$ kan skrivas om

som en linjär kombination av moderkrusningen $\psi(t)$ och skalningsfunktionen $\varphi(t)$ som

$$f(t) = c\varphi(t) + d_{0,0}\psi_{0,0}(t) + d_{0,1}\psi_{0,1}(t) + d_{1,1}\psi_{1,1}(t) + d_{1,2}\psi_{1,2}(t) + d_{2,2}\psi_{2,2}(t) + d_{2,3}\psi_{2,3}(t) + d_{3,3}\psi_{3,3}(t)$$

där alltså antalet termer motsvarar antalet definierade punkter i den ursprungliga funktionen. För att skapa transformen löses approximationskoefficienten c och detaljkoefficienterna $d_{n,k}$ ur uttrycket genom att beräkna ett lämpligt ekvationssystem. Resultatet brukar skrivas som en följd $(c, d_{0,0}, d_{0,1}, \ldots, d_{3,3})$ och i det här fallet blir det (4,5;−2;−1;−1;−1;−0,5;−0,5;−0,5;−0,5).

Nivå	C_a								C_d
3	4,5								−2
2	2,5				6,5				−1;−1
1	1,5		3,5		5,5		7,5		−0,5;−0,5;−0,5;−0,5
0	1	2	3	4	5	6	7	8	

Figur 11: Haartransformen för en kort serie

I det exemplet beräknades den fullständiga Haartransformen för serien, men den kan också beräknas steg för steg i flera nivåer. Transformen kan också beskrivas som en serie uträkningar av medeltal och differenser. Genom att utgå ifrån värdemängden för funktionen $f(t)$ från det föregående exemplet, följden (1;2;3;4;5;6;7;8), kan en Haartransform beräknas i flera steg, figur 11 illustrerar denna process. Nivå 0 betecknas som den ursprungliga serien tal. För att sedan räkna ut de följande nivåerna räknas parvis ett medeltal och differensen delad på två för alla tal ur den föregående nivån. I varje nivå kallas räckan med medeltal för approximationskoefficienter och betecknas C_a medan räckan med differenser kallas för detaljkoefficienter och betecknas C_d. För till exempel nivå 1 blir approximationskoefficienterna

$$C_a = \left(\frac{1+2}{2}; \frac{3+4}{2}; \frac{5+6}{2}; \frac{7+8}{2}\right)$$

och detaljkoefficienterna

$$C_d = \left(\frac{1-2}{2}; \frac{3-4}{2}; \frac{5-6}{2}; \frac{7-8}{2}\right).$$

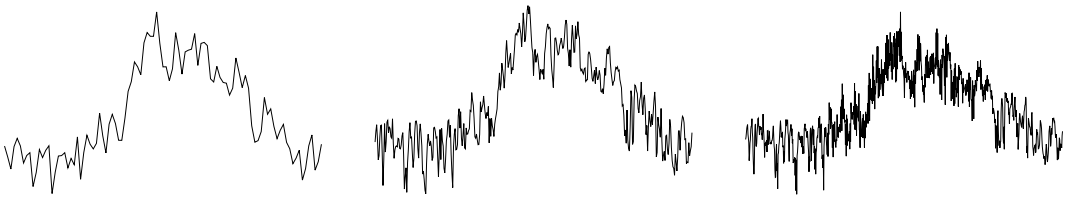
Ingen information försvinner när Haartransformen beräknas, med hjälp av detaljkoefficienterna kan alltid den föregående nivån beräknas genom att för varje approxima-

tionskoefficient först addera och sedan subtrahera detaljkoefficienten för att på det sättet återskapa två approximationskoefficienter ur den tidigare nivån.

Denna typ av transform kan användas som en metod för datareduktion genom att på det sättet som här visats transformera den ursprungliga serien tills den önskade nivån är nådd och sedan stoppa processen. Alternativt kan hela transformen beräknas och därefter kan en valfri nivå återskapas genom att svänga på processen.

I figur 9 illustreras hur en DWT-representation av flygplatsens hela temperaturserie ser ut vid olika upplösningar uträknade med Haartransformen. Intressant att se i figuren är att linjediagrammets utseende överlag bibehålls också i de lägre upplösningsnivåerna.

3.3 PAA

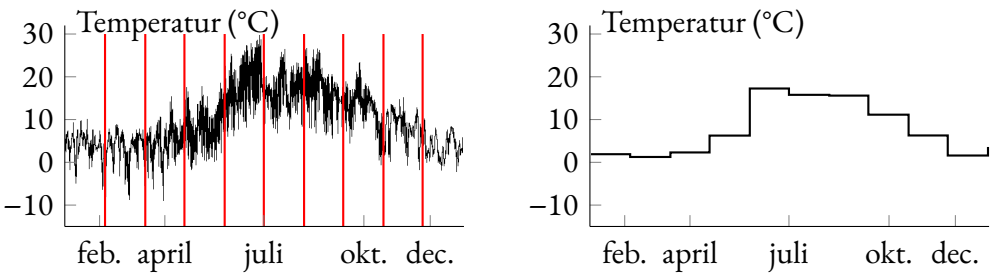


(a) 100 intervall (b) 400 intervall (c) 800 intervall

Figur 12: Figur 6 representerad med PAA i några upplösningsnivåer

Då ett linjediagram över en tidsserie ritas upp i en webbläsare, ritas det i praktiken oftast upp för att visas på en datorskärm som består av pixlar, dvs. så länge webbsidan inte skrivs ut. Pixelgittret på en skärm är diskret och varje pixel antar bara ett värde i taget. Då en tidsserie med många punkter, t.ex. den med lufttemperaturen år 2020 med ungefär 520 000 punkter, ritas upp som en bild med en upplösning på t.ex. 600 × 400 pixlar räcker inte pixlarna till för att varje punkt i tidsserien ska få en egen pixel, i praktiken kommer det finnas pixlar som flera gånger används och ritas över. I de fallen då resultatet i en pixel är det samma efter första överritningen som efter den sista är det klart att webbläsaren gjort ett onödigt arbete. Dessutom är också dessa punkter onödiga att överföra, det räcker med att skicka t.ex. den som skulle orsaka den sista förändringen i pixelns läge. Eftersom punkterna inte är identiska, de ritas i samma pixel på grund av upplösningen och inte på grund av deras egentliga värde, hjälper inte heller den sedvanliga komprimeringen som utförs när data skickas från webbservern. Det här motiverar följande fråga, finns det någon metod med vilken en tidsserie kunde förbehandlas så att inga ”onödiga” punkter överförs till webbläsaren?

Metoden *Piecewise Aggregate Approximation* (PAA) är en metod för datareduktion som utvecklats för tillämpning inom dataurvinning. För de datareduktioner som används i samband med dataurvinning ur tidsserier är målet att förenkla tidsserier på ett sätt som gör att de ändå kan jämföras efter förenklingen, dvs. två tidsserier som liknar varandra före



(a) Intervallindelning

(b) Förenklad tidsserie

Figur 13: Indelning i tio tidsintervall

datareduktionen ska också likna varandra efter datareduktionen, så att de kan lagras och indexeras i databaser. Målet för utvecklingen av PAA har varit att hitta en metod som är snabb att tillämpa men som ändå har denna viktiga egenskap att också de reducerade serierna kan jämföras. Metoden PAA tar alltså inte uttryckligen ställning till det visuella, men metoden är ändå intressant att undersöka, inte minst för att den kanske hör till de första metoder som en webbutvecklare kunde tänkas ta till för att förenkla en tidsserie. [23]

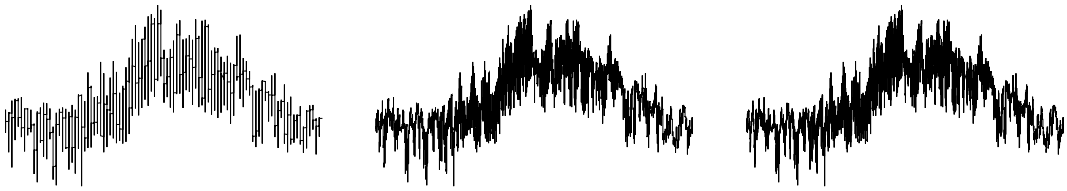
Med PAA sker datareduktionen genom att den ursprungliga tidsserien delas in i ett önskat intervall varefter ett aritmetiskt medelvärde beräknas för varje intervall. I Keogh m. fl. [23] definieras beräkningen som

$$\bar{x}_i = \frac{N}{n} \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} x_j$$

där n är antalet punkter i den ursprungliga tidsserien, N det önskade antalet punkter i den reducerade tidsserien $\bar{x}$ och x en tidsserie med samma definition som i ekvation 2. Det här uttrycket antar att N är en faktor i n, men i praktiken behöver detta inte heller vara fallet.

Denna följd av medelvärden $\bar{x}$ är alltså den tidsserierepresentation som metoden åstadkommer. Figur 13 illustrerar detta tillvägagångssätt med tio intervall. Det aritmetiska medelvärdet kan verka enkelt, men det har visat sig att tidsserier ofta uppvisar autokorrelation, dvs. mätningar som ligger nära varandra på tidsaxeln har ofta värden som också är nära varandra, och därför kan ett så enkelt mått som medelvärdet vara representativt för ett helt intervall.

I figur 12 är tidsserien för år 2020 representerad med hjälp av PAA i några olika antal intervall. Trenden för tidsserien verkar vara klar, också för de lägre antalen intervall. I figur 12 (c) börjar också de tätare områdena kring sommaren synas.



(a) 100 intervall (b) 400 intervall (c) 800 intervall

Figur 14: Figur 6 representerad med M4 i några upplösningsnivåer

3.4 M4

Datareduktionsmetoden M4 är uttryckligen utvecklad för att rita upp ett linjediagram över en lång tidsserie utan att diagrammet i sitt utseende avsevärt avviker från det ursprungliga. Detta åstadkommer M4 genom att skriva om databasförfrågningar så att dessa tar i beaktande upplösningen klienten har till sitt förfogande när den ritar upp ett diagram. Beskrivningen som följer är hämtad ur artikeln Jugel m. fl. [25] där också metoden presenterats med namnet M4.

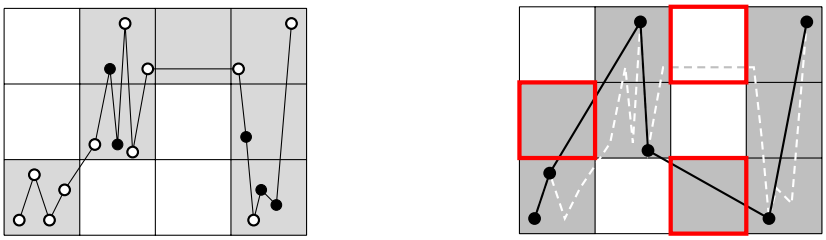
M4 är i första hand utvecklad för en pixelbaserad datorskärm, och räknar utöver detta med att linjediagrammet endast använder två färger, en för bakgrunden och en för linjen, så att varje pixel i diagrammet entydigt är antingen ifylld eller inte. Resultatet är ett linjediagram som till sitt utseende liknar det diagram som ritas upp med hjälp av hela datamängden men som i praktiken ritats upp med hjälp av en förenklad beskrivning av datan som därför snabbare överförs från server till klient.

Då ett linjediagram ritas upp på en datorskärm sker en rasteriseringsprocess i vilken linjen passas in på det rutmönster som skärmens pixlar utgör. Om antalet pixlar inte räcker till för att visa varje mätpunkt i data, till exempel så att en pixel motsvarar en mätpunkt, kommer flera punkter i mätdata att efter rasteriseringen representeras av samma pixlar på skärmen. Datareduktionen i M4 sker genom en process där databasförfrågningar skrivs om med tanke på den rityta som klienten använder för linjediagrammet. För att anpassa databasförfrågning kräver M4 två parametrar, båda mätta i pixlar, bredden w och höjden h på den rityta som klienten kommer att använda för att rita upp det resulterande linjediagrammet.

För att på rutnätet av pixlar projicera en högupplöst tidsserie delar M4 jämnt in tidsserien i w stycken intervall och väljer för varje intervall fyra punkter som får representera intervallet som helhet. Punkterna är de första och de sista punkterna i intervallet, $\min(t)$ och $\max(t)$, och det lägsta värdet och det högsta värdet i intervallet, $\min(v)$ och $\max(v)$. Detta förfarande påminner om algoritmerna som används i till exempel ljudediteringsprogram som Audacity, där den högupplösta vågformen förenklas med hjälp av topp- och bottenvärden[26].

Med hjälp av de fyra extremvärdena är det möjligt att rita upp en förenklad version av

tidsserien som på pixelnivå ändå ser exakt likadan ut som det diagram som skulle uppstå om varje punkt behandlades. Pixelkolumnens utsträckning längs med y-axeln kan nämligen räknas ut endast med hjälp $\min(v)$ och $\max(v)$ och för att räkna ut i vilka pixlar intervallen möts, vilket kan innebära ytterligare fyllda pixlar lägre ner eller högre upp längs med y-axeln $\min(v)$ och $\max(v)$, behövs bara $\min(t)$ och $\max(t)$. [25, del 4.3]



(a) Punkturval med M4

(b) Punkturval med min. och max.

Figur 15: Datarepresentation med M4 jämfört med min. och max.

Figur 15 är ett exempel hämtat ur Jugel m. fl. [25] och illustrerar motiveringen bakom tillvägagångssättet i M4. I figur 15 (a) är den ursprungliga serien uppritad som ett svart linjediagram. De med vitt ifyllda cirklarna är de punkter som väljs ut för en M4-representation och de svarta cirklarna är sådana som lämnas bort eftersom de inte påverkar det slutliga diagrammet. De gråa rutorna är de pixlar som fylls i då linjediagrammet ritas up. Figur 15 (a) illustrerar hur de punkter som i M4 väljs ut räcker för att rita upp ett linjediagram som täcker samma pixlar som den ursprungliga serien.

Figur 15 (b) fungerar som ett exempel på hur en enklare variant av metoden inte ger ett likadant resultat. I figuren syns den ursprungliga serien som en streckad linje och den förenklade serien som en heldragen linje. I figuren skapas en datareduktion genom att för varje intervall bara välja minimum och maximum, dvs. i jämförelse med M4 lämnas här bort de sista och första punkterna i varje intervall. I detta fall leder det till att två pixlar blir felaktigt ifyllda och en pixel som borde ha blivit ifylld lämnas bort. På det här sättet beskrivs M4 i Jugel m. fl. [25] som en lämplig metod för att med färre punkter exakt återge det ursprungliga diagrammet i en viss upplösning.

4 Digitala bilder

En digital bild består av en mängd pixlar. Varje pixel har en bestämd plats i bilden och ett tal som beskriver hur pixeln ser ut. Ett sätt att representera en gråskalebild är t.ex. att lagra ett tal mellan 0 (helt svart) och 255 (helt vit) som uttrycker vilken nyans av grå pixeln antar. En färgbild kan på ett liknande sätt representeras genom att använda RGB-modellen där varje pixel har ett värde som består av tre intensitetskomponenter, en röd, en grön och en blå. Förhållandet mellan komponenterna utgör färgblandningen och den slutliga färgen för pixeln. [20]

RGB-bilder kan lagras på flera sätt, till exempel så att varje pixels värde består av 24 bitar information där varje färg använder åtta bitar och följaktligen kan anta värden på skalan 0 (lägst intensitet) till 255 (högst intensitet). Digitala bilder kan lagras på många olika sätt, och det finns ett stort antal alternativ för hur färger kan hanteras. För de koncept som tas upp i det här arbetet är ändå dessa beskrivningar tillräckliga. [20]

4.1 Bilder som signaler

Digital bildbehandling är ett viktigt tillämpningsområde för signalbehandling. Bilder är flerdimensionella signaler och representeras därför ofta med hjälp av matriser eller funktioner som tar flera variabler som argument. Beroende på färgmodellen kan en bild vara en funktion av två eller tre variabler, video kan dessutom lägga till en ytterligare tredje (eller fjärde) tidsdimension. Dimensionen för en bild är alltså antalet koordinater som krävs för att identifiera värdet för en enskild pixel. [27]

Då datorer bearbetar matriser är det vanligt att en flerdimensionell matris överförs till en endimensionell radvektor som kan lagras i en enkel räcka i arbetsminnet. Efter att en bild överförts till denna endimensionella representation kan den behandlas som andra endimensionella signaler. Omvandlingen sker ofta i den s.k. rad-kolumn ordningen i vilken raderna i matrisen radas upp efter varandra i räckan. Enligt den här metoden kan t.ex. matrisen $\mathbf{A}$

$$\begin{bmatrix} a_{11} & \cdots & a_{1j} \\ a_{21} & \cdots & a_{2j} \\ \vdots & \ddots & \vdots \\ a_{(i-1)1} & \cdots & a_{(i-1)j} \\ a_{i1} & \cdots & a_{ij} \end{bmatrix}$$

omvandlas till radvektorn (räckan) $\mathbf{a}$

$$\begin{bmatrix} a_{11} & \cdots & a_{1j} & a_{21} & \cdots & a_{2j} & \cdots & a_{(i-1)1} & \cdots & a_{(i-1)j} & a_{i1} & \cdots & a_{ij} \end{bmatrix}.$$

4.2 MSE

Mean squared error (MSE) är inom signalbehandlingen ett mått på hur nära två signaler är varandra. MSE är ett av de vanligaste måtten, bl.a. på grund av att det är så enkelt att beräkna. MSE är medeltalet på kvadraterna av differenserna mellan två signaler och kan definieras som följande för signalerna $\mathbf{x} = \{x_1, x_2, \dots, x_N\}$ och $\mathbf{y} = \{y_1, y_2, \dots, y_N\}$. [28]

$$MSE(\mathbf{x},\mathbf{y}) = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2 \tag{8}$$

En bild som är identisk med den ursprungliga bilden har $MSE = 0$. Ju högre MSE desto mer olik är bilden den ursprungliga.

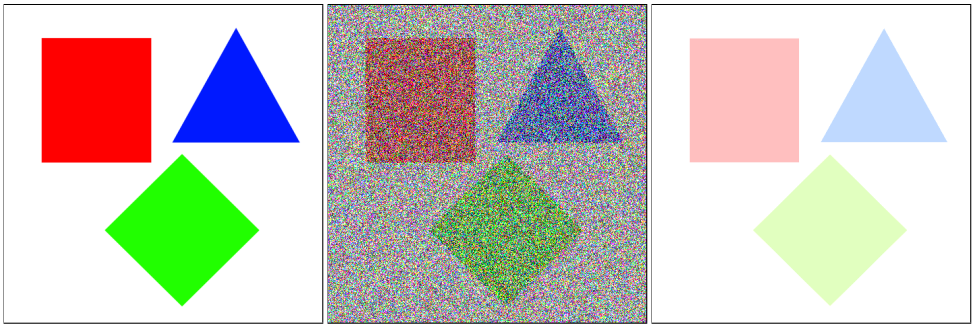
4.3 SSIM

Structural similarity index measure (SSIM) är ett längre utvecklat mått än MSE och tillämpar en modell som beskriver människans förmåga att se skillnader i bilder. Måttet har t.ex. använts för att utvärdera olika komprimeringsmetoder för digital video. [29]

SSIM beräknas i en flerstegsprocess där flera aspekter för bilderna tas i beaktande. I korthet mäts bildens luminans med hjälp av medelvärdet μ och kontrasten och strukturen med en kombination av varians σ och kovarians σ_{xy}. Resultatet är ett decimaltal i intervallet [0,1], där 0 står för en bild som inte alls liknar den ursprungliga och 1 är en bild som är representerad av exakt samma signal. Uträkningen sker i flera steg men kan förenklas till följande uttryck. [29]

$$SSIM(\mathbf{x},\mathbf{y}) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{({\mu_x}^2 + {\mu_y}^2 + C_1)({\sigma_x}^2 + {\sigma_y}^2 + C_2)} \tag{9}$$

Konstanterna C_1 och C_2 stabiliserar måttet då nämnaren närmar sig noll. De är definierade som $C_1 = (K_1 L)^2$ och $C_2 = (K_2 L)^2$ där K_1 och K_2 är två små konstanter, t.ex. 0,01 och 0,03, och L intensitetsomfånget för pixlarna, dvs. det största värdet som en pixel kan anta, t.ex. 255 för en 8-bitars gråskalebild. [29]



(a) MSE: 0 SSIM: 1 (b) MSE: 0,56 SSIM: 0,13 (c) MSE: 0,56 SSIM: 0,77

Figur 16: Hur MSE och SSIM och skiljer sig vid ett konstant fel

SSIM är speciellt utvecklat för att märka om ytor och former förändras på grund av en felsignal eller komprimeringsartefakter. Den här funktionen är illustrerad på ett konstgjort sätt i figur 16. Den första bilden, figur 16 (a), är den ursprungliga bilden. Figur 16 (b) är resultatet av en addition med syntetiskt brus och figur 16 (c) är resultatet av en addition av absolutbeloppet av samma brus. Den adderade signalen i figur 16 (b) är i varje pixel en slumpmässig subtraktion eller addition medan den i figur 16 (c) endast är en addition. MSE har samma värde i de båda förändrade bilderna eftersom magnituden på felet är samma. SSIM-värdet påverkas ändå av att bruset bryter upp de jämna ytorna i figur 16 (b) och gör att figur 16 (c) är närmare den ursprungliga [29]. Alla uträkningar av MSE och SSIM i det här arbetet är utförda med hjälp av Python-biblioteket *scikit-image* som beskrivs i artikeln Walt m. fl. [30]. Figur 16 är fritt efter ett liknande exempel som hittas i dokumentationen för *scikit-image*.

5 WWW och webbläsare

Webben (eng. *World Wide Web* eller helt enkelt *WWW*) är ett informationssystem på Internet som består av informationsstycken, s.k. resurser, med identifierare som kallas för URI:er. Webbsidor är resurser på webben som består av samlingar av HTML-dokument som förmedlas mellan server och klient med hjälp av protokollet HTTP. Webbsidor har ofta URI:er som kallas för URL, då identifieraren inte bara fungerar som ett namn utan också som instruktioner för hur resursen kan hittas i nätverket. [31]

Webbläsare kallas de programpaket som har i uppgift att läsa in webbsidornas HTML-dokument över HTTP-protokollet och att presentera innehållet för användaren. Förutom HTML innehåller dokumenten ofta också CSS-kod, som instruerar webbläsaren i hur innehållet ska presenteras, och program skrivna i programmeringsspråket JavaScript som kan göra webbsidan interaktiv. [32]

HTML är utvecklat för att skriva dokument. Webbläsare stöder ändå vid sidan om HTML ett flertal andra teknologier som kan fungera som byggnadsblock för att bygga upp interaktiva datavisualiseringar för att visas i en webbläsare. Till dessa byggnadsblock hör JavaScript, Ajax, SVG och JSON.

5.1 JavaScript

JavaScript är ett skriptspråk utvecklat för att lägga till interaktivitet i webbsidor. Språket tolkas i webbläsaren i samband med att sidan läses in och använder webbläsarens programmeringssgränssnitt för att göra ändringar i det innehåll som visas för användaren. Moderna webbsidor använder så gott som alla JavaScript i någon mån, enligt webbsidan W3Techs förekommer skriptspråket på hela 98 % av de 10 miljoner populäraste webbsidorna [33]. JavaScript är flexibelt och kan användas för allt från att hantera knapptryckningar och formulärcheckar till att hantera alla aspekter av webbsidan som s.k. webbapplikationer. JavaScript fick sin början i webbläsaren Netscape men är i nuläget standardiserat som en sammanslagning av egenskaper från flera tidiga skriptspråk för webbläsare. [34]

5.2 Ajax

En viktig funktion i moderna datavisualiseringar är förmågan att läsa in mer data efterhand. För en datavisualisering är det värdefullt att låta användaren enkelt gå från att studera en översikt av de data som samlats in till att titta på någon enskild detalj i visualiseringen. Moderna webbläsare stöder programmeringskonceptet Ajax där asynkrona funktionsanrop kan utföras i bakgrunden och inte kräver att resten av JavaScript-koden som körs i webbläsaren väntar på att t.ex. en HTTP-hämtning slutförs. Med hjälp av dessa kan data för datavisualiseringen läsas in i bakgrunden utan att störa användaren. [35]

5.3 SVG

Vektorformatet *Scalable Vector Graphics* (SVG) är ett format för ritningar och diagram som definieras med XML och tolkas i webbläsaren till bilder. Med termen vektor menas i det här fallet att bilderna inte beskrivs med hjälp av pixlar i ett på förhand bestämt koordinatsystem, utan de skapas som instruktioner till en SVG-tolk som tolkas i ett anpassningsbart koordinatsystem. Bilderna kan på det sättet visas vid flera olika upplösningar, dvs. i flera olika koordinatsystem, utan att de behöver skalas. [36, kap. 1]

SVG-diagram kan användas till flera ändamål. I webbläsare kan diagrammen bäddas in i HTML-dokumentet och deras utseende anpassas i takt med resten av webbsidan. Eftersom diagrammen blir en egentlig del av webbsidan kan de justeras och ändras med JavaScript också efter att webbsidan lästs in. Detta gör SVG-diagram speciellt användbara för interaktiva datavisualiseringar, ett diagram kan t.ex. innehålla förklarande text som först visas då användaren för sin muspekare över något intressant område och datapunkter kan markeras och färgas enligt vad användaren väljer att ska stå ut. [36, kap. 1]

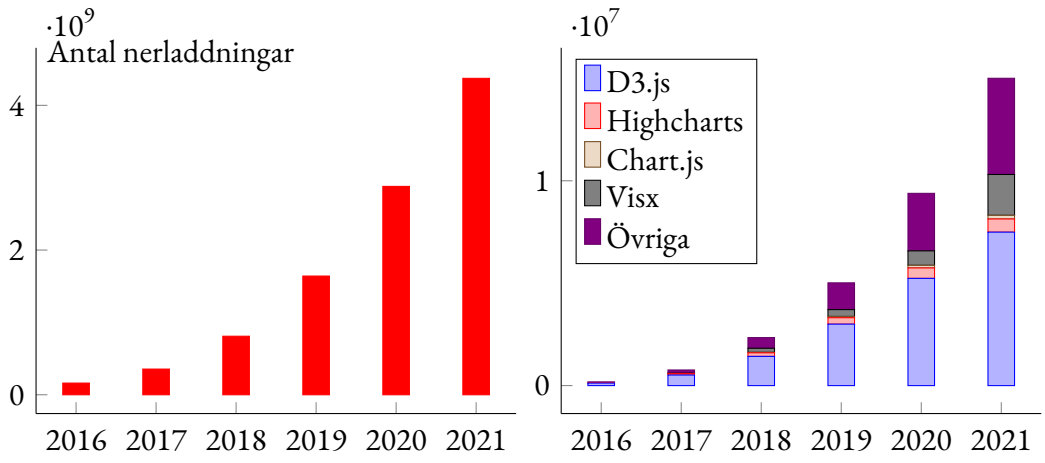
Förutom SVG stöder alla moderna webbläsare också Canvas-gränssnittet som kan användas för att rita upp pixelbilder med JavaScript. Med Canvas är det också ett möjligt att rita upp datavisualiseringar i webbläsare men gränssnittet kommer inte att behandlas i detta arbete. I allmänhet är Canvas-bilder mer beräkningseffektiva eftersom de behandlas på ett liknande sätt som bildfiler av webbläsare och inte som vektorgrafik. Detta betyder att en lösning för datareduktion som komprimerar data tillräckligt för att låta ett SVG-diagram ritas upp utan onödig fördröjning också borde fungera för en motsvarande Canvas-bild.

5.4 JSON

För att data skall kunna överföras mellan server och webbläsare krävs det att data som sparas på filsystemet omvandlas till ett format som både webbservern och webbläsaren kan läsa in och behandla. Ett format med utbrett stöd är formatet *JavaScript Object Notation* (JSON). JSON är ett textformat, data skickas alltså kodat som textsträngar. En viktig egenskap för datavisualiseringar är att JSON stöder flyttal med dubbel precision, dvs. tal som består av 64 bitar [37]. JSON låter också servern lätt inkludera metadata vid sidan om den egentliga datan, men den största fördelen med JSON är ändå att alla moderna webbläsare stöder formatet. En webbutvecklare kan vara säker på att JavaScript rutinerna som kör i slutanvändarnas webbläsare kan hämta de behövliga data från servern om de skickas som JSON.

5.5 Populära bibliotek för diagramritning

Ofta använder webb- och mjukvaruutvecklare sig av olika färdiga lösningar och programbibliotek för att implementera funktionalitet som utvecklaren kanske inte har rätt expertis eller tid för att implementera själv. De bibliotek som är i vanlig användning kan fungera som indirekta indikatorer för vilka typer av lösningar som slutanvändarna för tillfället är intresserade av. Här följer en kort undersökning med målet att visa att datavisualisering är något som många webbutvecklare arbetar med.



(a) Alla bibliotek

(b) Bibl. för datavisualisering

Figur 17: Npm-hämtningar per vecka i medeltal

I paketregistret Npms webbtjänst finns det ett flertal bibliotek som kan användas för att rita upp olika typer av diagram i webbläsaren. Det finns rentav så många av dessa programpaket att det kan vara svårt för en webbutvecklare att välja bland dem. För att göra det lättare för användaren har Npm en inbyggd funktion som rangordnar paketen enligt deras popularitet, ofta kan det hjälpa användaren att se vilka paket som andra verkar gilla. I Npm är ett pakets popularitet angiven som antalet gånger ett bibliotek laddas ner per dag, och sammanfattande statistik för detta kan hämtas som nerladdningar per vecka från en tredje part på webbsidan `npms.io`.[38]

Figur 17 illustrerar hur antalet nerladdningar av paket utvecklats sedan 2016, vilket är det tidigaste året för vilket det finns statistik att hämta. Figur 17 (a) visar hur antalet nerladdningar per vecka, sammanfattat per år, av alla programbibliotek som finns att hämta via paketregistret gått upp med åren. Sedan 2016 har nerladdningarna nästan femdubblats. I figur 17 (b) görs en indelning genom att endast visa utvecklingen för de 250 mest populära biblioteken för diagramritning och datavisualisering. Dessa 250 paket finns i paketregistret med sökordet *charts* eller sökordet *visualization*. De fyra biblioteken som nämns med namn i förklaringen är bland de mest populära. Eftersom dessa fyra är delar av stora utvecklingsprojekt är de uppdelade i många paket i registret. Här är alla de

relaterade paketen sammanfattade som en post per projekt. Utvecklingen ser ut att följa ungefär samma mönster som resten av biblioteken i registret, men i absoluta tal är antalet endast någon bråkdel av alla nerladdningar.

Ur denna statistik är det svårt att dra några konkreta slutsatser om hur användningen av paketen på nätet ser ut. Såvida inte den uppåtgående trenden är orsakad av någon förändring i hur statistiken är förd, verkar det ändå klart att Npm är en flera gånger mer populär tjänst än den var år 2016. Detta kan t.ex. vara ett resultat av att JavaScript som programmeringsspråk och webbläsaren som programmeringsomgivning blivit mera populära överlag. Det kan också vara resultatet av en centralisering kring paketregistret Npm, dvs. att paket som tidigare distribuerades på något annat sätt helt enkelt nu distribueras via Npm i stället. Trenden för paketen som är relaterade till datavisualisering verkar ändå peka uppåt. Om detta får fungera som en indikator för datavisualisering på webben i allmänhet, verkar det som om webbsidor också i framtiden kommer vara en viktig plattform för datavisualisering.

5.6 Interaktionsmetoder för datavisualisering i webbläsaren

Moderna webbläsare är så välutvecklade programpaket att det som begränsar de interaktionsmetoder som en webbutvecklare kan implementera närmast är webbutvecklarens egna kreativitet. För att kunna avgöra vilka interaktionsmetoder som skulle vara lämpliga att tillämpa kan det hjälpa att först kategorisera dem. En sådan kategorisering av interaktionsmodeller för datavisualisering presenteras i Yi m. fl. [39] där författarna har valt att dela in interaktionsmetoderna i sju kategorier. Kategorierna består av olika sätt att:

1. *välja* intressanta data.
2. *utforska* data genom att går från ett parti till ett annat.
3. *omfördela* data genom att t.ex. sortera eller arrangera data.
4. *omkoda* data genom att byta typen av visualisering.
5. *abstrahera/utarbeta* data som visualisering på allmän resp. detaljerad nivå.
6. *filtrera* bort data enligt något kriterium.
7. *ansluta* datapunkter som är relaterade till varandra.

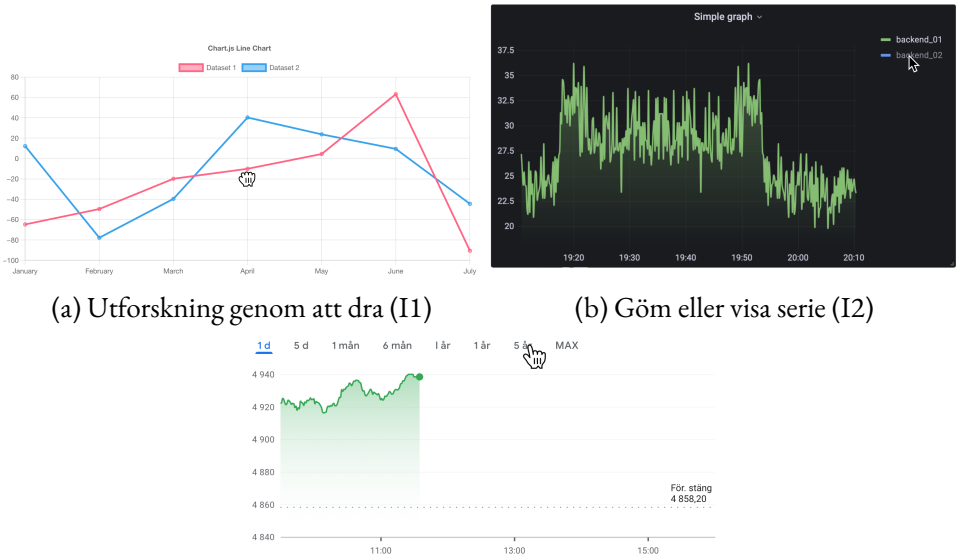
En uttömmande diskussion som skulle behandla hur dessa olika kategorier förekommer på webben är inte målet med detta arbete. Tre interaktionsmetoder som påträffas på webben betraktas ändå här som speciellt intressanta för linjediagram över tidsserier:

I1 utforskning av data genom förflyttning av vyn i sidled (eng. *pan*).

I2 filtrering av data genom att gömma eller visa serier.

I3 inläsning av mera data efter hand.

Med kategorierna ur Yi m. fl. [39] kan dessa tre i ordningsföljd kategoriseras under *utforska*, *filtrera* och *abstrahera/utarbeta*.



(a) Utforskning genom att dra (I1)

(b) Göm eller visa serie (I2)

(c) Läs in mera data efter hand (I3)

Figur 18: Några vanliga interaktionsmetoder på nätet

De tre sätten illustreras i figur 18 som skärmavbilder av hur de i några fall ser ut i praktiken. Figur 18 (a) är ett exempel på hur ett sätt att förflytta visualiseringsfönstret i sidled längs med tidsaxeln (interaktionsmetod I1) är implementerat i diagramritningsbiblioteket Chart.js. Figur 18 (b) visar hur dashboardverktyget Grafana hanterar en filtrering av serier som redan är inlästa i webbläsaren (I2) och figur 18 (c) visar hur aktieportalen Google Finance stöder vyer av data på detaljnivåerna ett dygn, fem dygn, en månad osv. (I3).

Dessa tre sätt att interagera med datavisualiseringar påträffas i praktiken, men det finns som sagt också många andra koncept för hur interaktion sker som kunde behandlas. För detta arbete räcker det att dessa tre fungerar som exempel för en diskussion kring hur en interaktiv datavisualisering bereder särskilda utmaningar för en framställning av tidsserier.

6 Problembeskrivning

Tidsserier är ofta långa och består av data insamlade över en lång tid. Därför är de ofta också intressanta att undersöka både på en allmän trendnivå, t.ex. för att se förändringar över år eller decennier, och på en låg detaljnivå, t.ex. för att se dygnsvariation. För detta ändamål passar linjediagram, och en lockande datavisualisering är interaktiv genom att låta användaren själv välja ut de tidsintervall som är intressanta. På det sättet kan användaren på kort tid utforska data och använda datavisualiseringen som stöd för att utarbeta en noggrannare analys.

En interaktiv visualisering av denna typ av data är ändå utmanande. Datamängden kan vara för stor för att kunna läsas in i arbetsminnet som en helhet och för en visualisering i en webbläsare kan överföringstiden från server till klient vara ett hinder för att alla data visas på samma gång. Som det nämndes i det föregående kapitlet kan också moderna webbläsare med JavaScript hämta in data bitvis i bakgrunden, men det kan vara utmanande att bestämma precis vilka data som borde hämtas i bakgrunden. Diagram i SVG-formatet fungerar väl för interaktiva visualiseringar, de stöder animationer och kan justeras med JavaScript, men deras flexibilitet innebär också att stora SVG-filer är både mer arbetsminnes- och processortidskrävande än bildfiler med samma innehåll. Målet med det här kapitlet är att definiera några av de utmaningar som en interaktiv visualisering av tidsserier på webben bereder.

6.1 Tumregler för interaktivitet

En långsam webbsida är inte trevlig att använda. Hur väl en långsam webbsida tolereras beror bl.a. på vad användaren vill åstadkomma, hur länge användaren redan klickat runt på sidan och hur användaren tänker sig att systemet fungerar [40]. Tumreglerna för hur långsam en webbsida, eller ett datorgränssnitt i allmänhet, får vara beror enligt Nielsen [41] på vilket intryck utvecklaren vill ge åt användaren. För en interaktion där användaren ska få intrycket att denne direkt manipulerar någonting på datorn, t.ex. flyttar på en fil från en mapp till en annan genom att plocka upp och dra den med muspekaren, är tumregeln att systemet ska reagera inom 0,1 sekunder. Då användaren utforskar t.ex. filsystemet eller klickar runt mellan webbsidor är tumregeln 1 sekund. I det här fallet förstår användaren att datorn gör beräkningar för att producera innehållet för nästa steg, men genom att hålla väntetiden till som mest 1 sekund blir övergången mellan t.ex. olika dokument och mappar smidig och användaren känner sig ännu ha kontrollen över vad som händer på datorn. För att användaren inte ska tappa koncentrationen och glömma bort vad målet med handlingen är ska datorn kunna producera något resultat inom 10 sekunder. Många användare lämnar helt enkelt en webbsida om det tar längre än 10 se-

kunder att läsa in den, men också upprepade väntetider på endast några sekunder medan användaren utforskar webbsidan kan leda till att användaren blir frustrerad och ger upp.

6.2 Utmaningar för tidsserier på webben

De tre tumreglerna för reaktionstider är användbara för flera sorters applikationer, bland annat för interaktiva datavisualiseringar på t.ex. följande sätt. För en tidsserie som ritas upp som ett linjediagram kan det vara önskvärt att användaren kan förstora något intressant parti av linjen. Om användaren sedan vill jämföra med någon annan region i diagrammet kan det kännas naturligt att användaren kan använda muspekaren för att ta tag i linjen och dra den i sidled för att flytta tidsfönstret antingen framåt eller bakåt i tiden. Detta motsvarar interaktionsmetod I1 som behandlades i förra kapitlet. För att åstadkomma en sådan handling bör SVG-diagrammet animeras och justeras enligt den första tumregeln, så att en förändring syns inom 0,1 sekunder. Då får användaren en känsla att tidsfönstret faktiskt följer muspekaren. Denna typ av interaktivitet är enkel att implementera med SVG men för ett stort diagram med många punkter kan det inte alltid gå att förlita sig på att handlingen kan utföras inom den utsatta tiden.

För den andra storleksordningen av händelser som borde behandlas inom en sekund är t.ex. filteroperationer som hanterar data som redan är inlästa i webbläsaren. Om t.ex. flera tidsserier visas i samma linjediagram och de enskilda linjerna kan väljas bort eller läggas till och om detta inte sker tillräckligt snabbt kan det vara svårt för användaren att använda funktionen för att jämföra flera serier. Detta omfattar den andra interaktionsmetoden, I2. Att bara visa och gömma SVG-linjer går att utföra relativt snabbt, men om tidsserierna måste läsas in från servern eller beräknas efterhand kan långa tidsserie göra att användarupplevelsen haltar.

De handlingar som faller under den tredje storleksordningen är datahämtning och den tid som krävs för att rita upp det första diagrammet. Här förstår användaren troligen att servern och webbläsaren kan behöva lite tid för att åstadkomma den önskade visualiseringen och en tidsanvändningen på upp till tio sekunder kan vara acceptabel. Den tredje interaktionsmetoden I3 illustrerar detta fall. Om användaren hamnar vänta längre än 10 sekunder kan något verka vara på tok, att data inte går att läsa in. HTTP-förfrågning som tar mycket längre än 10 sekunder kan också i praktiken vara så lång att webbläsaren avbryter den i tron att servern inte kan svara. Om en lång tidsserie tar för lång tid att läsa in kan en effektiv lösning vara att bara hämta in det tidsintervall som användaren har valt att undersöka. Om detta bitvis inhämtande av data ändå är märkbart långsamt kan användaren ändå i längden blir frustrerad om det handlar om en handling som upprepas flera gånger.

För tidsserier är mängden data mycket beroende på hur stort tidsintervallet är som undersöks. En optimal lösning skulle tillåta att alla handlingarna som utgör en interaktiv

datavisualisering skulle gå att utföra inom konstant tid, dvs. oberoende mängden data som behandlas. I praktiken är en sådan lösning utmanande att hitta, t.ex. den vanliga databasen PostgreSQL som används i detta arbete använder s.k. B-träd för sina index vilket betyder att en sökning efter data i databasen har en tidskomplexitet som i bästa fall är $O(\log n)$. Genom att utföra en datareduktion på servern kan ändå mängden data som överförs till webbläsaren hållas inom samma storleksordning för en viss upplösningsnivå. Om en lämplig datareduktionsmetod hittas kan de ovannämnda handlingarna som går ut på att data manipuleras i webbläsaren med säkerhet implementeras så att användaren får se resultat inom de utsatta tiderna. Då återstår att metoden för datareduktion väljs så att den också gör behandlingen på servern tillräckligt effektiv för att också t.ex. dataöverföringen i de flesta fall kan utföras inom den utsatta tiden på tio sekunder.

I resten av arbetet undersöks de fyra datareduktionsmetoderna DFT, DWT, M4 och PAA som beskrivits tidigare. Målet är att se i vilken mån dessa löser de utmaningar som här presenterades, samt att se om jämförelsen av metoderna kan leda till insikt kring hur en datareduktion i allmänhet borde planeras för tillämpning i en webbapplikation.

7 Testsystem

Som det framgick ur det föregående kapitlet finns det många utmaningar med att rita upp interaktiva linjediagram över tidsserier i en webbläsare. I detta arbete är undersökningen begränsad till att omfatta fyra situationer som kan leda till en dålig användarupplevelse:

1 Webbsidan och diagrammets data hämtas för långsamt från servern

2 Diagrammet skapas för långsamt i webbläsaren och insättningen i HTML-dokumentet dröjer för länge

3 Diagrammet är för stort och interaktiviteten haltar

4 Diagrammet beskriver inte den ursprungliga tidsserien

Målet med arbetet är att undersöka hur de fyra metoderna för datareduktion som beskrivits i kapitel 3 kunde lösa problemen ovan. Meningen är att utreda både kvantitativa och kvalitativa aspekter för metoderna genom att besvara några frågor.

Till de rent kvantitativa frågorna hör följande:

T1 Mätt i millisekunder, hur lång tid tar det att läsa in de nödvändiga data från servern?

T2 Mätt i millisekunder, hur lång tid tar det för JavaScript-programmet att skapa ett SVG-diagram på basis av data och rita upp det i webbläsaren?

T3 Mätt i kilobyte, hur stor är den textsträng som SVG-linjen utgör (korrelerar med minnesanvändning)?

T4 Enligt MSE, liknar bilden den ursprungliga?

T5 Enligt SSIM, liknar bilden den ursprungliga?

De kvalitativa testerna ska besvara följande frågor:

T6 Är metoden enkel att implementera, eller finns det en färdig implementation som går att använda?

T7 Passar implementationen in i den vanliga arkitekturen för webbapplikationer?

T8 Löser metoden interaktivitetsproblemet?

T9 Helt subjektivt, liknar det nya diagrammet det ursprungliga?

7.1 Testkomponenter

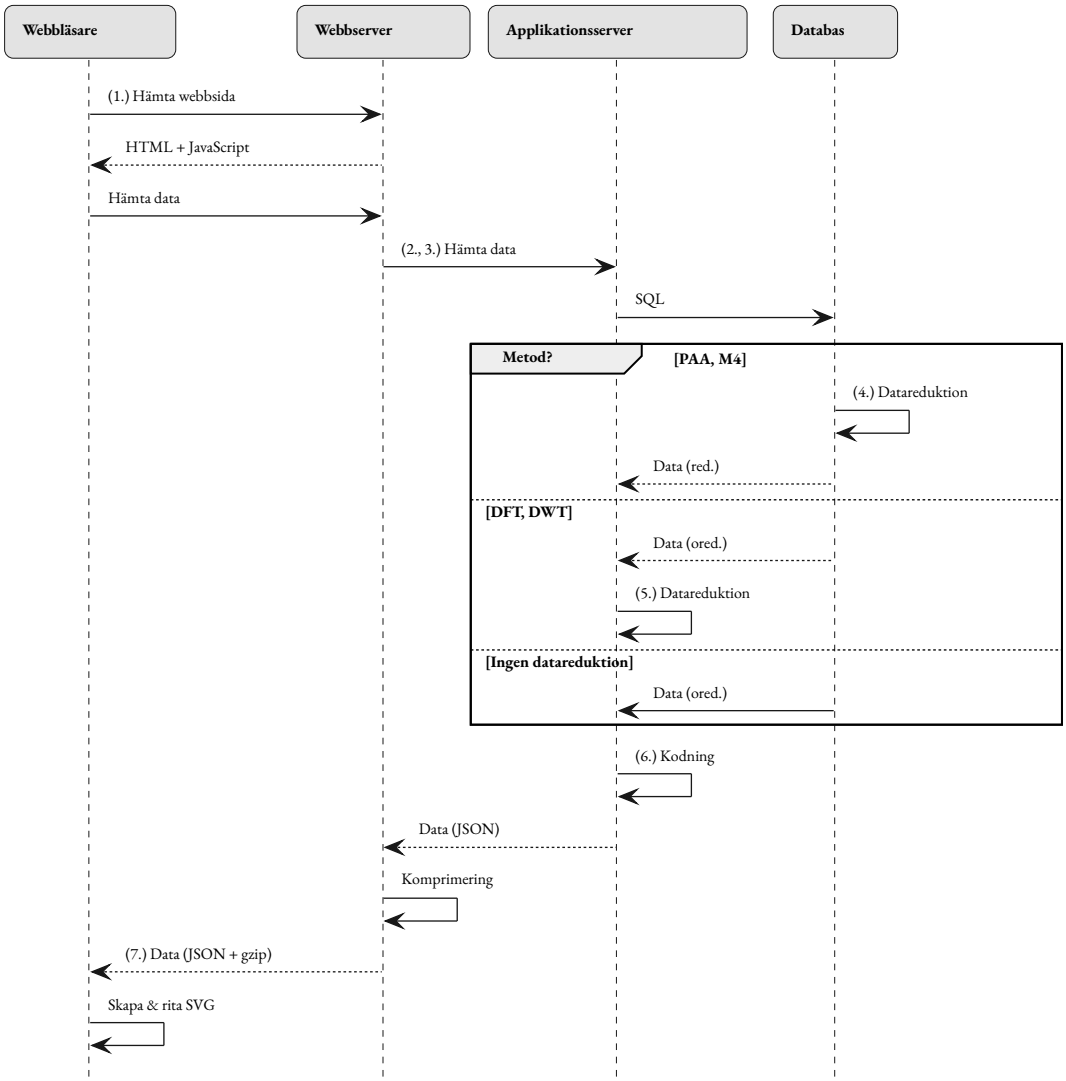
Jämförelserna i detta arbete är utförda på en vanlig persondator, en bärbar dator av modell Macbook Pro 17,1 som kör operativsystemet Mac OS 12.2. Datorn har en processor med 8 kärnor och en klockfrekvens på upp till 3,2 GHz och flashbaserat lagringsutrymme.

För att undersöka metoderna simuleras en typisk omgivning för en dynamisk webbapplikation där huvudkomponenterna är en klient och en server. Klienten är den i Mac OS inbyggda webbläsaren Safari och den är automatiserad med hjälp av testomgivningen Selenium. Servern består i sin tur av tre delar, en webbserver, en applikationsserver och en databas.

Då båda parterna körs på samma dator motsvarar inte nätverksförbindelsen de vanliga omständigheterna på webben, i stället för att data skickas över Internet skickas de i detta fall över det s.k. länklokala nätverket på datorn där endast testdatorns resurser påverkar överföringshastigheten. På detta sättet är de enskilda testkörningarna mer jämförbara sinsemellan, nätverkets bandbredd förändras inte mellan testerna, inga noder faller bort eller kommer till på paketens väg genom nätverket och inga nätverksfel inträffar. De uppmätta tiderna kan, och ska, alltså jämföras sinsemellan men detta arbete tar inte ställning till hur de kunde användas för att förutspå hur överföringstiderna ser ut i praktiken.

Sekvensdiagrammet i figur 19 visar hur en förfrågning behandlas i testsystemet. Då webbläsaren hämtar data för att rita upp ett diagram sker följande:

1. Webbläsaren hämtar webbsidan och skickar en förfrågning för att hämta data.
2. Webbservern tar emot förfrågning, märker att den inte kan fullfölja den själv och skickar den vidare till applikationsservern.
3. Applikationsservern översätter parametrarna ur förfrågning till en SQL-förfrågning som den sedan skickar till databasen.
4. Då datareduktionsmetoden är implementerad som SQL sker datareduktionen redan i databasen och databasen svarar applikationsservern med den reducerade tidsserien, i annat fall svarar databasen helt enkelt med alla data inom det aktuella tidsintervallet.
5. Om datareduktionsmetoden är implementerad på applikationsservern tillämpas sedan datareduktionen på den ursprungliga tidsserien.
6. Efter att tidsserien reducerats kodar applikationsservern om data till JSON.
7. JSON-texten skickas därefter tillbaka till webbservern som komprimerar den med en s.k. *lossless* komprimeringsmetod, som alltså inte tappar information, och skickar till slut den som svar till webbläsaren.



Figur 19: Flödet i testsystemet

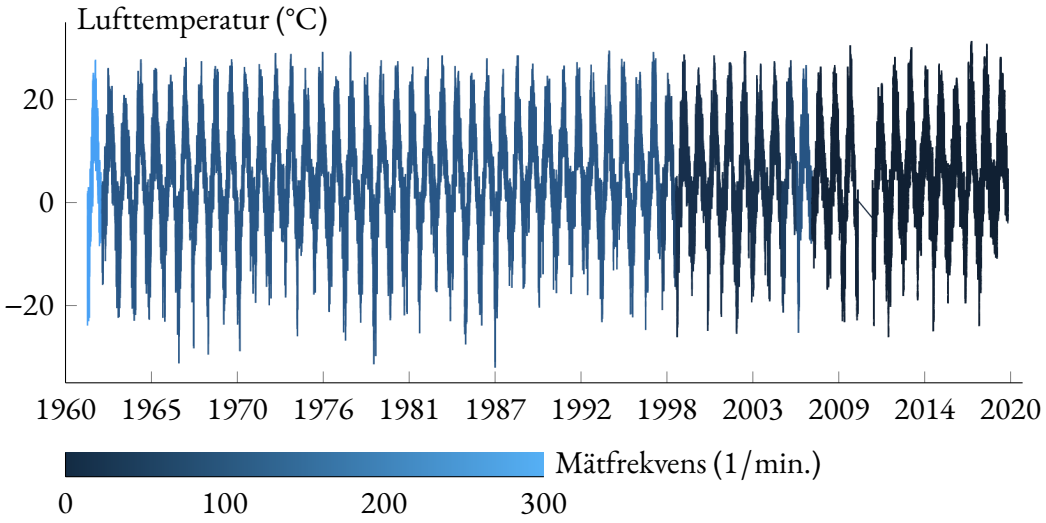
Servern består i detta arbete av webbservern Nginx, applikationsservern Gunicorn och PostgreSQL som databas. På Gunicorn körs ett Python-program som implementerar arbetets tester som databasförfrågningar och numeriska beräkningar. Tabell 1 räknar upp programversionerna för de komponenter som använts.

7.2 Data

För att studera hur de olika metoderna beter sig i praktiken krävs lämpliga data, för detta arbete är målet att ha så många mätpunkter i tidsserierna att uppritningsprocessen blir svårhanterlig och resultatet på det sättet för en slutanvändare inte optimalt. Tidigare i detta arbete har olika diagramtyper och datareduktionsmetoder illustrerats med hjälp av lufttemperaturen på Åbo flygplats. Fördelen med att arbeta med temperaturmätningar är att de är lätta att förstå och kontrollera. Mycket kalla temperaturer i juli är t.ex. sannolikt ett tecken på att något är fel i systemet eller till och med ett mätfel. Lufttemperaturen

Tabell 1: Programpaket och versioner

Komponent	Programvara	Version
Webbserver	Nginx	1.21.6
Applikationsserver	Gunicorn	20.1.0
Databas	PostgreSQL	14.2
Programpaket för numeriska beräkningar	NumPy	1.22.2
Programpaket för beräkningar med krusningar	PyWavelets	1.2.0



Figur 20: Lufttemperaturen och dess mätfrekvens vid Åbo flygplats åren 1960–2020

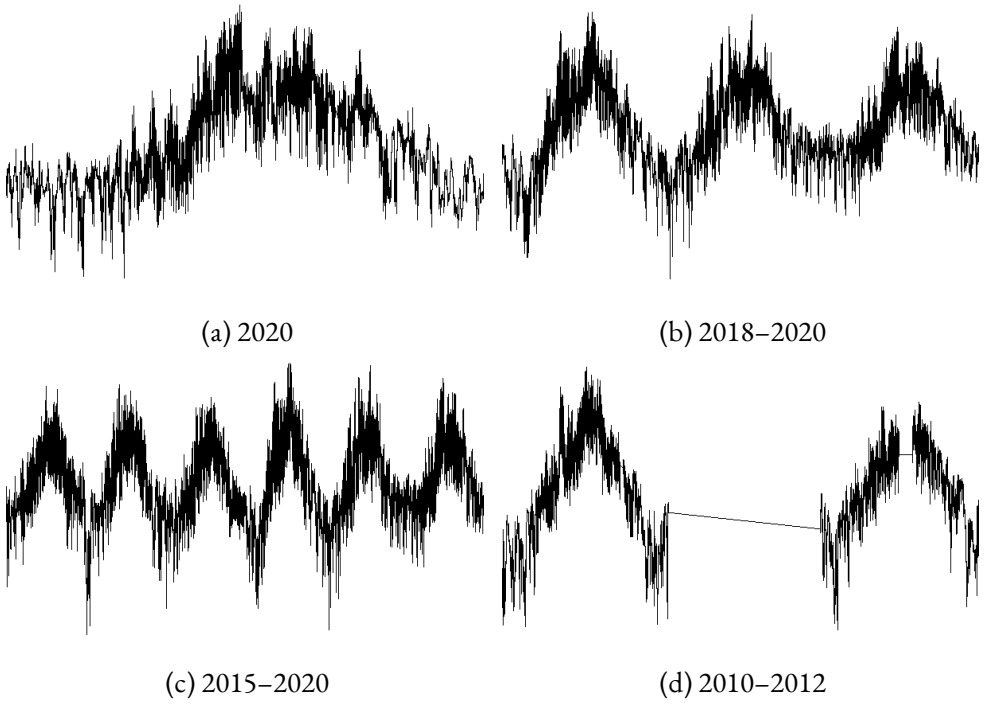
stiger och sjunker relativt ofta, men ändå på en sådan skala att de är enkla för människan att urskilja. Temperaturmätningarna vid Åbo flygplats är också utförda så ofta att uppsättningarna data är tillräckligt stora för att kräva någon form av förbehandling innan de ritas upp.

Via Meteorologiska institutets nerladdningstjänst har nästan alla de temperaturmätningar som finns tillhanda för Åbo flygplats hämtats. De äldsta mätningarna är från år 1960 och de nyaste har blivit begränsade till år 2020. [19]

Hur ofta temperaturen antecknats, och därigenom det totala årliga antalet temperaturmätningar, har varierat från år till år. Under tiden 1960–1997 har temperaturen antecknats ungefär var tredje timme, vilket betyder drygt 2900 temperaturmätningar per år. Under tiden 1998–2007 är variationen större, men i regel verkar temperaturen ha blivit uppskriven varje timme, dvs. ungefär 8600 gånger per år. Åren 2008–2010 är mängden mätningar ungefär 50 000 per år, vilket innebär att mätningar utförts var tionde minut. Från och med år 2012 finns temperaturmätningar att hämtas på minutbasis, dvs. ungefär 520 000 temperaturmätningar per år. Av någon anledning fattas år 2011 helt och hållet förutom en halv månad i januari. År 2012 saknar också ungefär en månad. Perioderna där temperaturen mätts med största regelbundenhet är åren 1961–1997 och 2013–2020.

Det totala antalet temperaturmätningar för tiden 1960–2020 är kring 5 miljoner efter att ungefär 5000 tomma mätningar plockats bort.

Figur 20 illustrerar temperaturutvecklingen som ett linjediagram, och variationen i tid mellan mätningar är uppritad med hjälp av en färgskala där den ljusaste färgen motsvarar ett mellanrum mellan mätningar på dryga fem timmar och den mörkaste färgen ett mellanrum på kring en minut.



(a) 2020 (b) 2018–2020

(c) 2015–2020 (d) 2010–2012

Figur 21: Referensdiagram för fyra testintervall

Som det syns i figur 20 kan det totala antalet mätningar verka stort. Om mätningarna skulle presenteras i ett annat format, t.ex. utskrivna på en lång rulle papper i en hög upplösning, kan de nästan fem miljoner punkterna kanske vara rentav nödvändiga. Huruvida några uppsättningar data är för stora för att hantera eller visa upp beror alltså på användningsområdet.

Hela intervallet 1960–2020 kunde användas för att jämföra metoderna, men det stora antalet mätpunkter kunde tänkas vara en speciellt stor utmaning för metoderna DFT och DWT då det handlar om en icke-konstgjord signal, dessutom är det intressant att se hur metoderna beter sig med några olika intervall. I detta arbete har fyra olika testintervall valts, år 2020, åren 2018–2020, åren 2015–2020 och åren 2010–2012. Exempel på hur mätningarnas tidsserier ser ut uppritade som linjediagram syns i figur 21.

Avståndet mellan mätningarna är i alla intervall förutom för åren 2010–2012 i medeltal en minut med endast några enstaka avvikelser. För att undersöka hur metoderna beter sig vid data som inte är lika systematiskt uppmätta har intervallet 2010–2012 valts ut som ett intervall där tiden mellan olika mätpunkter varierar mer än i resten av intervallen. År

2010 är avståndet mellan varje mätpunkt i medeltal 10 minuter, för år 2011 fattas alla mätningar förutom halva januari och inom år 2012 var avståndet mellan varje mätpunkt i medeltal en minut på samma sätt som i de mer regelbundna intervallen, men juli månad fattas nästan helt och hållet. Här kan noteras att glappen i data syns i figur 21 (d) som bara visar data ur det glappa intervallet medan endast mellanrummet på platsen för år 2011 syns i den tidigare figuren över hela intervallet 1960–2020.

7.3 Testparametrar

För att undersöka de frågor som räknas upp i början av detta kapitel skickas upprepade förfrågningar igenom testsystemet med olika parametrar.

De parametrar som systemet stöder är:

1. Metod för datareduktion (ingen, PAA, M4, DFT, DWT)
2. Bredd i pixlar på den rityta som webbläsaren använder för SVG-diagrammet
3. Upplösningsnivå som den ursprungliga tidsserien ska reduceras till
4. Tidsintervall för data

Ett test delas upp i två skeden, i det första mäts tiderna för de viktiga momenten i processen och i det andra skedet jämförs en PNG-bild av det SVG-diagram som webbläsaren ritat upp med en referensbild som skapats med samma parametrar men utan någon tillämpad datareduktion.

Tabell 2: Testresultat för en uppritning av linjediagrammet över år 2020 med hjälp av metoden PAA

(d) Tidsanvändning

Steg	Tidsanvändning
Dataurval	120 ms
Databeh. i DB	64 ms
DB till Python	0 ms
Datareduktion i Python	0 ms
JSON-kodning	0,82 ms
Dataöverföring	1 ms
JSON-avkodning	2 ms
SVG-konstruktion	1 ms
DOM-insättning	1 ms
Totalt	190 ms

(d) Mått

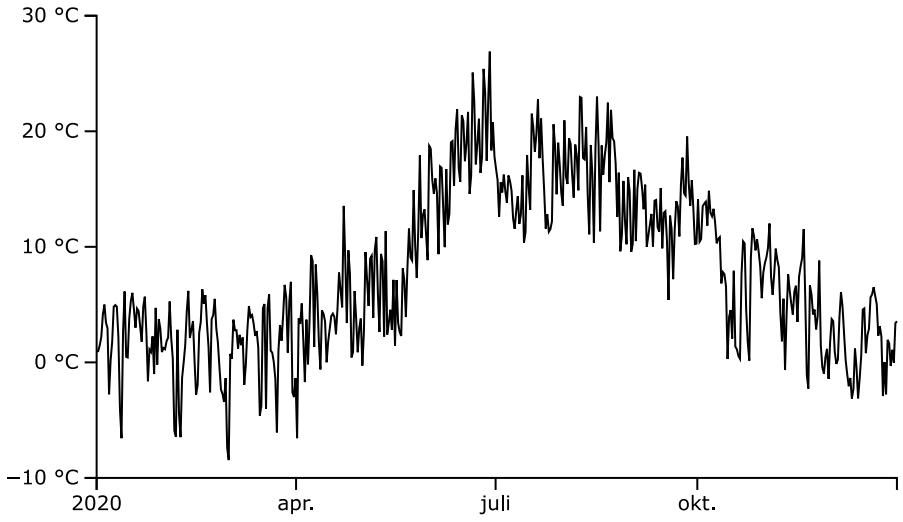
Mått	Värde
Överförda punkter	599
Datamängd	47 kB
Mätpunkter innan red.	522 357
Diagrambredd	640 px
Punkter i förfrågning	600
SVG-storlek	15 kB
MSE	5614
SSIM	0,78

Som ett exempel syns resultatet av en testkörning i tabell 2. Testkörningen gjordes med metoden PAA och upplösningsnivån 600 över år 2020. I tabell 2 (d) visas tidsanvändningen i några viktiga skeden och i tabell 2 (d) syns några detaljer för bl.a. tidsserien och resultatet i bildjämförelsen.

Med upplösningsnivå menas här en parameter som alla fyra metoder stöder men på olika sätt. Den behöver inte heller alltid överensstämma med diagrammets upplösning i pixlar och inte heller med antalet punkter i den reducerade tidsserien. För metoden PAA innebär upplösningsnivån det antal intervall som den ursprungliga tidsserien ska delas in i. Med metoden DFT kan upplösningsnivån varieras genom att justera antalet koefficienter som bibehålls då den inverterade transformen beräknas. I dessa båda fall blir resultatet en tidsserie med ett antal punkter som är lika med upplösningsnivån. För metoden M4 innebär upplösningsnivån också det antal intervall som tidsserien delas in i, men metoden föreskriver ändå att antalet punkter i resultatet ska vara fyra gånger det antal intervall som valts. Med metoden DWT kan inte antalet punkter i den reducerade tidsserien justeras jämnt utan hoppar i takt med att transformens nivå väljs.

Till näst följer en mera detaljerad beskrivning om hur var och en av metoderna i detta arbete är tillämpad och hur deras respektive upplösningsnivåer kan väljas.

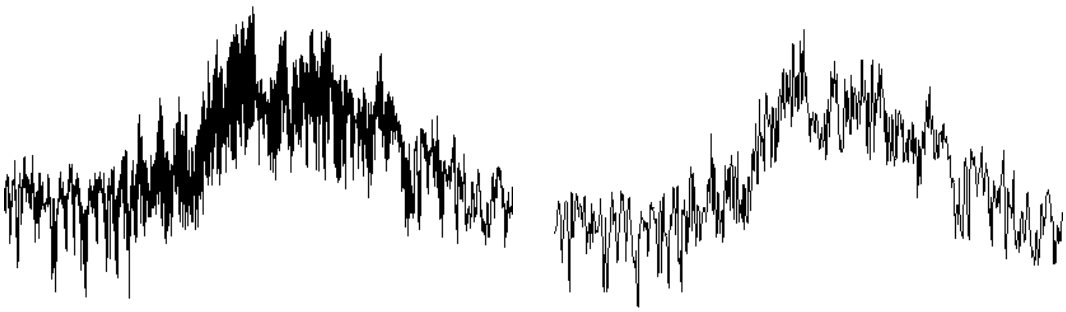
7.4 Visualiseringen



Figur 22: Linjediagram med D3.js

För att rita upp ett linjediagram i webbläsaren används i detta arbete JavaScript-biblioteket D3.js som hjälper användaren genom att hantera koordinatsystemet. Genom att använda biblioteket räcker det att definiera ett domän för båda axlarna och en utsträckning i pixlar på webbsidan för att skapa ett koordinatsystem som kan användas för att sedan rita upp en SVG-linje. Resultatet illustreras i figur 22.

För att kunna göra en jämförelse med SSIM krävs två bilder. I detta fall skapas för varje testintervall först en referensbild genom att rita upp ett SVG-diagram över tidsserien utan någon datareduktion varefter sedan testbilderna i de olika upplösningsnivåerna i tur och ordning ritas upp. Målet är att denna bild inte ska vara komprimerad och utan kant-



(a) Skärmbild av ett referensdiagram (b) Skärmbild av ett reducerat diagram

Figur 23: Två som PNG avbildade linjediagram

utjämning, dvs. helt enkelt en tvåfärgs bild där varje pixel antingen är färgad svart eller vit. Detta åstadkoms genom att för SVG-diagrammet sätta attributet *crispEdges* som ber webbläsaren att stänga av kantutjämning för diagrammet. För att spara en bild av diagrammet används webbläsarens funktion att ta en skärmbild av en nod i dokumentet och bildformatet PNG utan en alfakanal. Detta ger en bild som inte är komprimerad och bara består av svarta och vita pixlar. För bilderna lämnas också allt annat i diagrammet utom linjen bort. Figur 23 visar hur två sådana skärmbilder med en bredd på 400 pixlar ser ut.

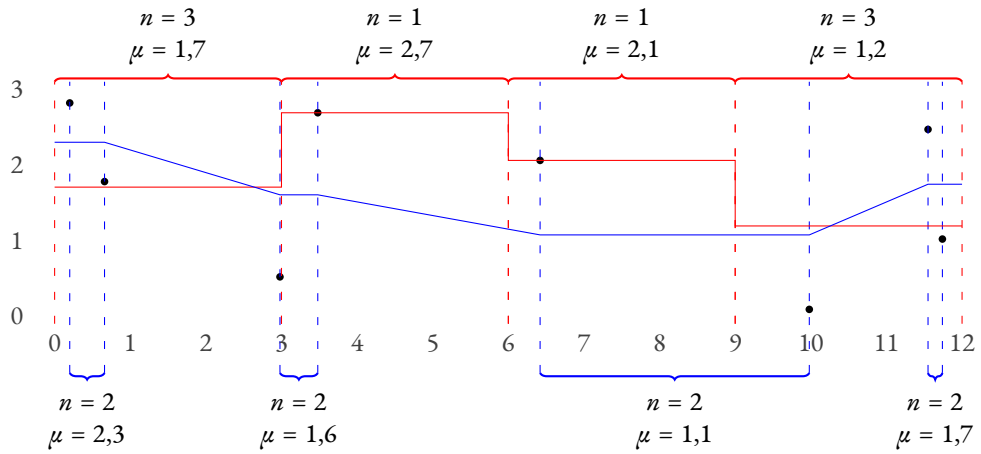
7.5 PAA

Metoden PAA är enkel att implementera, men i Keogh m. fl. [23] föreskrivs ingen särskild tillämpning av metoden. Den grundläggande idén för denna metod är som redan tidigare diskuterades att dela in en tidsserie i N lika stora intervall där varje intervall innehåller n element, och för varje intervall räkna ut intervallets medelvärde. Om längden på tidsserien inte är delbar med det önskade värdet på N måste serien förkortas så att inget intervall har en längd olika n.

För en endimensionell tidsserie, dvs. en sådan där endast ordningen av punkterna är angiven, kan reduktionen tillämpas med matriser och matrisprogramvara eftersom varje intervall har samma längd.

En tidsserie med längden K, t.ex. $\mathbf{x} = \{x_1, x_2, x_3, \ldots, x_K\}$, kan i radordning ordnas om till en matris $\mathbf{X}$ enligt följande mönster med n och N definierade som ovan. För att sedan beräkna medeltalet i varje intervall beräknas radvis i matrisen en summa och sedan divideras summan med antalet element. Resultatet av datareduktionen är alltså kolumnvektorn $\overline{\mathbf{x}}$ med ett uträknat medeltal för varje rad i $\mathbf{X}$.

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & \ddots & & \\ \vdots & & \ddots & \\ x_{N1} & & & x_{Nn} \end{bmatrix}, \overline{\mathbf{x}} = \begin{bmatrix} \frac{1}{n}\sum_{i=1}^{n} x_{1i} \\ \frac{1}{n}\sum_{i=1}^{n} x_{2i} \\ \vdots \\ \frac{1}{n}\sum_{i=1}^{n} x_{Ni} \end{bmatrix}$$



Figur 24: Intervallindelning

Matrisrepresentationen är praktisk för beräkningar i programpaket som Matlab och Octave som innehåller procedurer som snabbt kan utföra en radvis summa och division om alla rader, dvs. intervall i den ursprungliga serien, är lika långa. För en flerdimensionell tidsserie där tidpunkten för en mätning också anges är denna representation däremot inte alltid möjlig.

För glesa tidsserier, sådana där mätpunkterna inte är uppmätta med jämna mellanrum, kan det vara viktigt att också tidsintervall utan mätningar syns i datavisualiseringen. Det kan också vara viktigt att intervallen inte sträcker sig över ett för stort område i tiden.

En möjlighet för att inte ta mätningarna ur sina tidskontexter är att tillåta olika stora grupper och istället dela in data i N lika långa tidsintervall. Detta leder ändå till att datareduktionen inte kan tillämpas med hjälp av matrisberäkningar eftersom raderna kan bli ojämnt långa.

Denna uppdelning har också andra nackdelar, bl.a. att antalet punkter i de olika intervallen kan variera drastiskt och leda till medelvärden som inte är jämförbara. Det har ändå visats att punkter som ligger nära varandra i tiden i allmänhet också ligger nära varandra vad gäller deras värden, då är det möjligt att medelvärdet till och med kan fela mindre än om det tar i beaktande punkter som är långt avskilda i tiden [23].

Figur 24 illustrerar hur representationen kan skilja beroende på hur serien delas upp. I

bilden är en gles tidsserie med åtta punkter uppmätta över ett tidsintervall på 12 tidpunkter uppritad. I de blåa intervallen är serien uppdelad i fyra intervall med två mätpunkter i varje och i de röda intervallen är serien uppdelad i fyra intervall med fyra tidpunkter i varje. De blåa intervallen hör alltså till en indelning i fyra intervall med lika många *mätpunkter* i varje medan de röda intervallen är en indelning med lika många *tidpunkter* i varje. Båda sätten att dela in intervall har också fått varsin linje dragen genom medelvärdenas punkter och linjerna avviker märkbart från varandra. Figuren illustrerar alltså vikten av att noggrant välja vilket sät som intervallindelningen görs på.

För detta arbetes ändamål är PAA och M4 implementerade med hjälp av en SQL-förfrågning i databasen och implementationen använder det senare sättet att gruppera, dvs. det där tidsintervallet delas upp i flera mindre tidsintervall som sinsemellan är lika stora. På detta sätt blir implementationen mer jämförbar med de andra metoderna då detta är hur tidsserien delas upp för dem, och dessutom fungerar denna intervallindelning bättre för den glesa tidsserien i testintervallet 2010–2012. Se kodlistning 1 bland bilagorna för den SQL-förfrågning som implementerar metoden.

7.6 M4

Grundidén för M4 liknar den föregående metoden PAA och är följaktligen implementerad på ett liknande sätt. Intervallindelningen görs genom att dela in tidsseriens tidsutsträckning i intervall på samma sätt som beskrevs för metod PAA och den motsvarar på detta sätt tillämpningen i artikeln Jugel m. fl. [25] i vilken M4 presenterades. SQL-förfrågningen avviker ändå lite från det exempel som presenterades i artikeln eftersom den likt de andra tillämpningarna använder sig av funktionen *date_bin* som är unik för PostgreSQL och inte ingår i SQL-standarden. Se kodlistning 2 bland bilagorna för tillämpningen som använts i detta arbete.

7.7 DFT

Datareduktionen med hjälp av DFT är implementerad med hjälp av Python-biblioteket NumPy (beskrivet i t.ex. Harris m. fl. [42]) som innehåller färdiga funktioner för att beräkna en DFT.

Tillvägagångssättet är det att servern först hämtar hela tidsserien inom ett visst tidsintervall från databasen för att sedan beräkna en DFT och en inverterad DFT för tidsserien. I inverteringssteget reduceras data genom att endast de N första koefficienterna ges till den inverterande funktionen, dvs. det antal som motsvarar den önskade upplösningsnivån. Detta ger en serie med N mätpunkter.

Genom denna process faller tidsinformationen för varje mätpunkt bort men ordningen bibehålls. För att återskapa den delas tidsintervallet in i N tidpunkter på samma sätt

som i tillämpningarna för metoderna PAA och M4 och därefter tilldelas varje mätpunkt i ordning en tidpunkt varefter datareduktionen är färdig.

7.8 DWT

Datareduktion med DWT är implementerad med hjälp av Python-biblioteket PyWavelets (beskrivet i t.ex. Lee m. fl. [43]) som innehåller implementationer för olika krusningar och funktioner med vilka en DWT kan beräknas och inverteras. [43]

7.8.1 Val av krusning

Det finns många olika krusningar att välja mellan. Valet av krusning hänger nära ihop med hur signalen ser ut, om den har ”spetsiga” partier etc. För att fatta ett beslut om vilken krusning som skulle användas i detta arbete har alla krusningarna som PyWavelets biblioteket erbjuder jämförts med varandra enligt sina respektive SSIM- och MSE-värden i förhållande till en referensbild. Krusningarna har jämförts genom att rita upp linjediagram bestående av ungefär 1000 mätpunkter i en 600 pixlar bred SVG som ett förenklat linjediagram över år 2020.

Krusningarna ger rent subjektivt liknande bilder, men genom att jämföra bilderna kvantitativt blir ändå en viss skillnad klar. Tabell 3 listar de tio krusningarna ordnade enligt SSIM-värde för den resulterande bilden. Bland de med högsta SSIM verkar Haarkrusningen vara den mest lämpade, men Biorthogonal 2.2 har ändå ett lägre (dvs. bättre) värde för MSE. Daubechies 2 ger i sin tur ett SSIM-värde snarlikt det som Biorthogonal 2.2 ger, men också ett högre MSE. Eftersom Haarkrusningen i praktiken beräknar samma sak som PAA (se t.ex. Keogh m. fl. [23]) studeras i detta arbete DWT-metoden med hjälp av krusningen Biorthogonal 2.2. Denna krusning är inte likt Haarkusningen definierad som ett matematiskt uttryck utan endast som den filteruppsättning som tillämpar transformen. För att illustrera denna typ av krusningar brukar istället en approximation av funktionen beräknas. Se figur 25 för en illustration av krusningen som används i detta arbete.

7.8.2 Upplösningsnivåer

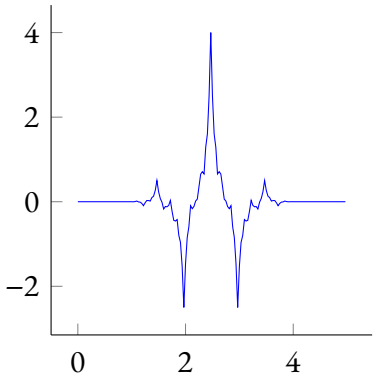
Metoden DWT består som sagt av en transform i flera nivåer där följande nivå beräknas med hjälp av den föregående. Då målet är att skapa en reducerad tidsserie med N punkter måste en passande nivå för transformen väljas. I följande ekvationer betecknas en avrundning neråt till närmaste heltal med $\lfloor \cdots \rfloor$ och en avrundning uppåt med $\lceil \cdots \rceil$.

För tillämpningen av DWT som används i detta arbete ges antalet punkter efter en transform av uttrycket

$$n = \lfloor \frac{K + a}{2} \rfloor \tag{10}$$

Familj	Krusning	SSIM	MSE
Haar	-	0,7959	4320
Daubechies	2	0,7887	4501
Biorthogonal	2.2	0,7886	4262
Biorthogonal	3.1	0,7868	4288
Symlets	4	0,7857	4565
Biorthogonal	1.3	0,7836	4658
Coiflets	2.1	0,7830	4719
Daubechies	3	0,7812	4905
Reverse biorthogonal	1.3	0,7805	4898
Biorthogonal	3.3	0,7801	4576

Tabell 3: De tio krusningarna med högsta SSIM



Figur 25: Moderskrusning för Biorthogonal 2.2

där K är det totala antalet punkter i den serie som transformeras och $a = d - 1$ med d som längden på det filter som implementerar den aktuella krusningen [43]. Detta gäller alltså för en övergång från en nivå till nästa. Utgående från detta uttryck kan antalet punkter på en valfri nivå l beräknas som en expansion av en geometrisk serie, dvs. med faktorn $q = \frac{1}{2}$ blir uttrycket

$$n_l = \lfloor \frac{K}{2^l} + \frac{a(1 - q^{l+1})}{1 - q} - a \rfloor. \tag{11}$$

Det exakta antalet punkter är ändå inte alltid intressant. Eftersom det önskade antalet punkter lätt hamnar mellan två DWT-nivåer är frågan vilken nivå som kommer närmast det önskade antalet punkter. I detta arbete väljs alltid den högre nivån, dvs. den nivå som ger färre punkter än det önskade antalet så att antalet punkter inte går över den ”budget” för punkter som webbläsaren indikerar att den har.Eftersom det dominerande fenomenet är att antalet punkter mer eller mindre halveras för varje ny nivå i transformen (se ekvation 10) kan en passande nivå också beräknas med hjälp av den binära logaritmen och ett önskat antal punkter n_o enligt

$$l = |\lceil \log_2 n_o - \log_2 K \rceil|. \tag{12}$$

I följande exempel är talen avrundade till två decimaler för att spara utrymme. Som ett exempel, det totala antalet mätpunkter i temperatureserien för 2020 är $K = 523\,796$. Med krusningen Biorthogonal 2.2 är längden på filtret $d = 4$. För en transform till nivå $l = 9$ blir antalet punkter enligt ekvation 11 följande.

$$
\begin{aligned}
n_l &= \lfloor 523796/2^4 + (3(1 - (1/2)^{10})/(1 - (1/2)) - 3 \rfloor \\
n_l &= \lfloor 1000 + 6 - 3 \rfloor \\
n_l &= 1000
\end{aligned}
$$

Om den önskade upplösningsnivån ska innehålla 1000 punkter, dvs. $n_o = 1000$, kan nivån hittas med hjälp av ekvation 12 som

$$
\begin{aligned}
l &= ||\lceil \log_2 1000 - \log_2 523796 \rceil| | \\
l &= ||\lceil 10 - 19 \rceil| | \\
l &= ||\lceil -9 \rceil| | \\
l &= 10
\end{aligned}
$$

vilket enligt ekvation 12 motsvarar ett antal punkter $n_l = 514$. Detta är en försiktig algoritm som ger en upplösningsnivå som innehåller färre punkter än det önskade 1000

8 Resultat

För att undersöka hur valet av upplösningsnivå och tidsintervall påverkar resultat av datareduktionen har alla fyra metoder tillämpats på data vid flera upplösningsnivåer. För att undersöka metoderna har en datareduktion vid upplösningsnivåer som utgör jämna hundratal ur intervallet [100, 6000] beräknats för varje metod och resultatet ritats upp i en webbläsare där diagrammet fått pixelbredden 600. De uppmätta testresultaten är alla avrundade till två gällande siffror eftersom den precisionen räcker för de jämförelser som utförs i detta arbete. För SSIM-måttet är förstås decimalerna viktiga, det är en stor skillnad mellan $SSIM = 0{,}80$, $SSIM = 0{,}90$ och $SSIM = 1$, men för de övriga mätningarna är det närmast skillnaden i storleksordning som är avgörande.

Målet har varit att se om en upplösning för datareduktion som matchar bredden på diagrammet är tillräcklig för en lämplig bild, och att se hurdant resultat metoderna ger då upplösningen skärps utöver detta.

Tabell 4 sammanfattar resultatet för de fyra datareduktionsmetoderna i de fem kvantitativa teststegen som beskrivits tidigare, dvs. tiden som det tar att överföra data från servern till webbläsaren (T1), tiden som det tar att i webbläsaren rita upp diagrammet (T2), storleken i bytes på den SVG-fil som utgör diagrammet (T3), hur långt diagrammet är ifrån den ursprungliga illustrationen enligt MSE (T4) och hur nära diagrammet är det ursprungliga enligt SSIM (T5). För varje metod och testskede visar tabellen ett medeltal som är uträknat över alla upplösningsnivåer och de fyra testintervallen 2020, 2018–2020, 2015–2020 och 2010–2012 (se figur 21 i föregående kapitel).

Tabell 4: Resultat för de kvantitativa testskedena beskrivna i kapitel 7

Metod	Överföring[T1]	Uppritning[T2]	SVG-storlek[T3]	MSE[T4]	SSIM[T5]
Basfall	4500 ms	520 ms	53 000 kB	0	1
PAA	580 ms	3,8 ms	95 kB	3800	0,86
M4	1000 ms	8,2 ms	380 kB	760	0,97
DFT	2000 ms	4,5 ms	96 kB	5600	0,80
DWT	1900 ms	4,9 ms	150 kB	5000	0,81

Medeltalen sammanfattar alltså många upplösningsnivåer. Eftersom de flesta testkörningar använt relativt höga upplösningsnivåer, endast en tiondel med samma eller lägre antal intervall som pixelbredden, beskriver medeltalet närmast hur metoderna beter sig vid höga upplösningsnivåer. De tre första testintervallen 2020, 2018–2020, 2015–2020 är som tidigare förklarats relativt likadana, i alla tre fall är tiden mellan varje mätning en minut. Det fjärde fallet 2010–2012 innehåller några diskontinuerliga punkter där avståndet mellan mätningarna är avsevärt längre än en minut.

Tabellen visar att alla fyra metoder kan vara snabbare än basfallet i att hämta data från databasen och att överföra mätningarna till klienten (test T1), fastän de också under denna tid utför en datareduktion. Oberoende av vilken datareduktionsmetod som väljs leder också de färre punkterna till en snabbare uppritning av linjediagrammet i webbläsaren. Förbättringen för alla de fyra metoderna är i samma storleksordning, och är i alla de fyra fallen avsevärd jämfört med basfallet (test T2). De kortare tidsserierna hjälper också att minska på storleken på den textsträng som utgör linjediagrammets SVG-linje, i alla de fyra fallen är strängen mycket kortare (test T3). För metoderna DWT och M4 blir SVG-strängen längre än för de övriga metoderna eftersom antalet punkter i DWT och M4 inte alltid överensstämmer med antalet intervall i upplösningsnivån. Jämförelsen av bildkvalitet (testerna T4 och T5) visar förväntade värden för basfallet ($MSE = 0$ och $SSIM = 1$) och klart annorlunda värden för de fyra datareduktionsmetoderna.

Resultatet kommenteras närmare i följande avsnitt, men utgående från tabell 4 framgår det redan att metoden PAA överlag är den snabbaste metoden och att DFT är den långsammaste. I stort verkar metoden M4 ge det resultat som närmast motsvarar den ursprungliga bilden och DFT det resultat som är längst ifrån referensen. I de följande avsnitten kan tabellerna som visas jämföras med resultatet i tabell 5 som alltså innehåller resultatet för referensfallet i testintervallen.

Tabell 5: Referenstest för basfallet

Intervall	Överföring[T1]	Uppritning[T2]	SVG-storlek[T3]	MSE[T4]	SSIM[T5]
2020	1700 ms	190 ms	19 000 kB	0	1
2018–2020	4900 ms	560 ms	57 000 kB	0	1
2015–2020	9900 ms	1100 ms	110 000 kB	0	1
2010–2012	1700 ms	190 ms	19 000 kB	0	1

Eftersom alla de fyra metoderna åstadkommer en datareduktion löser de interaktivitetsproblemet i test T8, dvs. det att punkterna i den ursprungliga serien är för många för att webbläsaren ska klara av ett interaktivt SVG-diagram för tidsserien. De reducerade tidsserierna lämpar sig också för att överföras snabbt till webbläsaren och på det sättet kan användaren snabbare växla mellan olika tidsintervall och -serier. Resultatet i de övriga kvalitativa testen sammanfattas i tabell 6 och kommenteras i de följande avsnitten.

Tabell 6: Resultat för de kvalitativa testskedena beskrivna i kapitel 7

Metod	Praktisk[T6]	WWW-anpassad[T7]	Interaktiv[T8]	Liknar[T9]
Basfall	Ja	Delvis	Nej	–
PAA	Ja	Ja	Ja	Ja
M4	Delvis	Ja	Ja	Ja
DFT	Ja	Delvis	Ja	Delvis
DWT	Delvis	Delvis	Ja	Delvis

8.1 PAA

Tabell 7 visar resultatet i de fem kvantitativa testerna för PAA. Tabellen är uppdelad enligt de fyra testintervallen som beskrivits tidigare. För varje intervall och test är ett medelvärde uträknat över alla testkörningar.

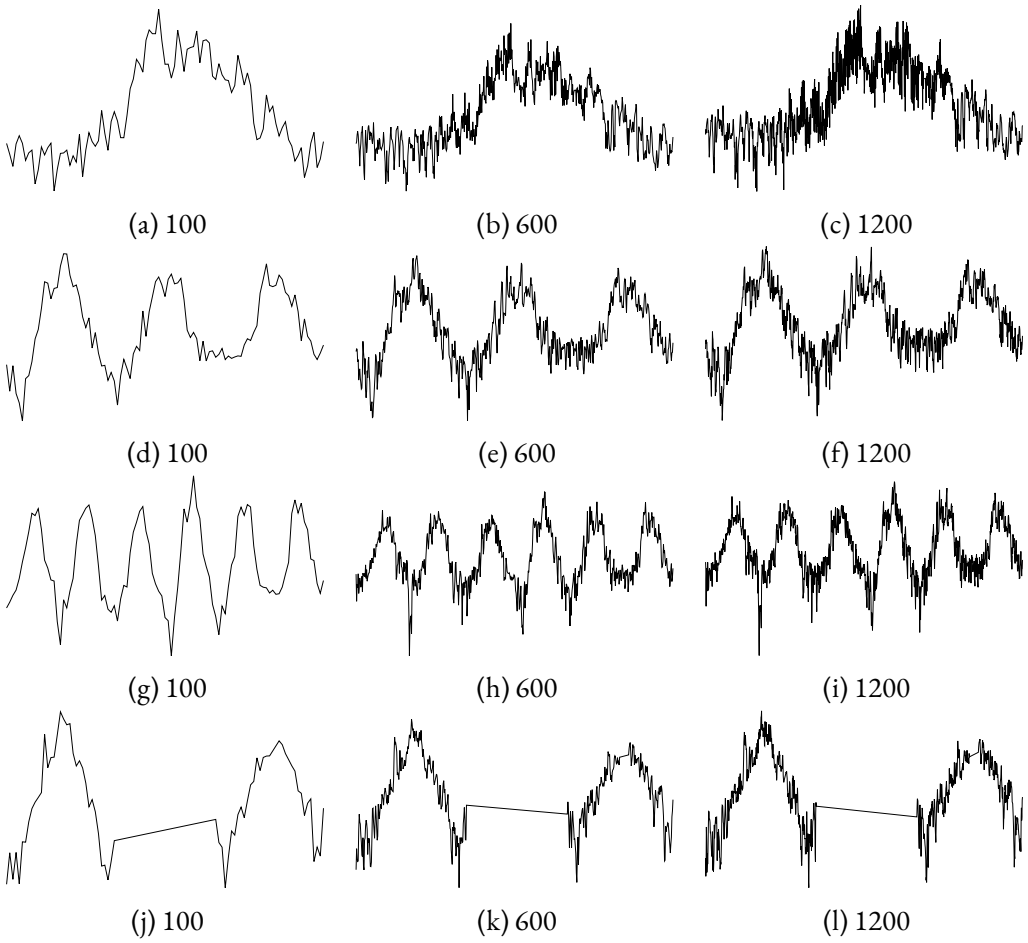
Tabell 7: Resultat för PAA

Intervall	Överföring[T1]	Uppritning[T2]	SVG-storlek[T3]	MSE[T4]	SSIM[T5]
2020	200 ms	3,9 ms	100 kB	2700	0,89
2018–2020	640 ms	4,2 ms	100 kB	4400	0,84
2015–2020	1300 ms	3,8 ms	100 kB	5500	0,81
2010–2012	200 ms	3,3 ms	68 kB	2900	0,89

Resultatet för de olika intervallen följer samma mönster som resultatet överlag som beskrivs i tabell 4, dvs. på grund av den reducerade tidsserien som metoden åstadkommer blir både SVG-diagrammet mindre (test T3) och överföringstiden och tiden som webbläsaren tar för att rita upp diagrammet (testerna T1 och T2) avsevärt kortare än för referensfallet. För MSE- och SSIM-testen visar tabellen att bildkvaliteten blir sämre ju längre tidsintervallet blir, dvs. ju fler mätpunkter som metoden har att behandla.

Gällande de kvalitativa testerna har det blivit klart att metoden är enkel att tillämpa, test T6 i detta testsystem. Versionen av databasen PostgreSQL som använts i detta arbete stöder SQL-funktionen *date_bin* som automatiskt delar in en tidsserie i önskat antal intervall. Med denna funktion kan metoden tillämpas också på en oregelbunden tidsserie med endast en kort SQL-förfrågning. Eftersom metoden är enkelt att använda med en SQL-databas passar den också in i en typisk webbapplikation (test T7).

Med figur 26 och referensfiguren 21 ur det föregående kapitlet kan test T9 undersökas, dvs. diagrammen kan jämföras på en helt subjektiv nivå. Figuren består av utdrag ur de bilder som under testkörningarna ritats upp i det 600 pixlar breda testkoordinatssystemet. Jämfört med referensbilden verkar metoden PAA leda till linjediagram som bibehåller den överhängande trenden. Extremvärden faller ändå bort, och den utjämnande effekten



Figur 26: Det visuella resultatet i några upplösningsnivåer med PAA

av att varje intervall är lika värt leder till att informationen om vilka partier av tidsserien som uppmätts tätare faller bort.

8.2 M4

Likt PAA följer också resultatet för M4 de mönster som redan den sammanfattande tabell 4 visar. Metoden M4 leder till en avsevärt snabbare uppritning av linjediagrammet i webbläsaren och en SVG-bild som tar upp mycket mindre arbetsminne och skivutrymme än referensfallet. Denna metod leder också till en avskalad tidsserie som trots en längre behandling i databasen än referensfallet leder till en snabbare överföring av linjediagrammet. Enligt tabell 8 verkar överföringstiden öka ju längre den ursprungliga tidsserien är. För testintervallet 2015–2020 uppgår överföringstiden till så mycket som 2200 ms vilket är ungefär fem gånger kortare än referensfallet i samma intervall (se tabell 5) men ändå en väntetid som torde vara märkbar för en användare.

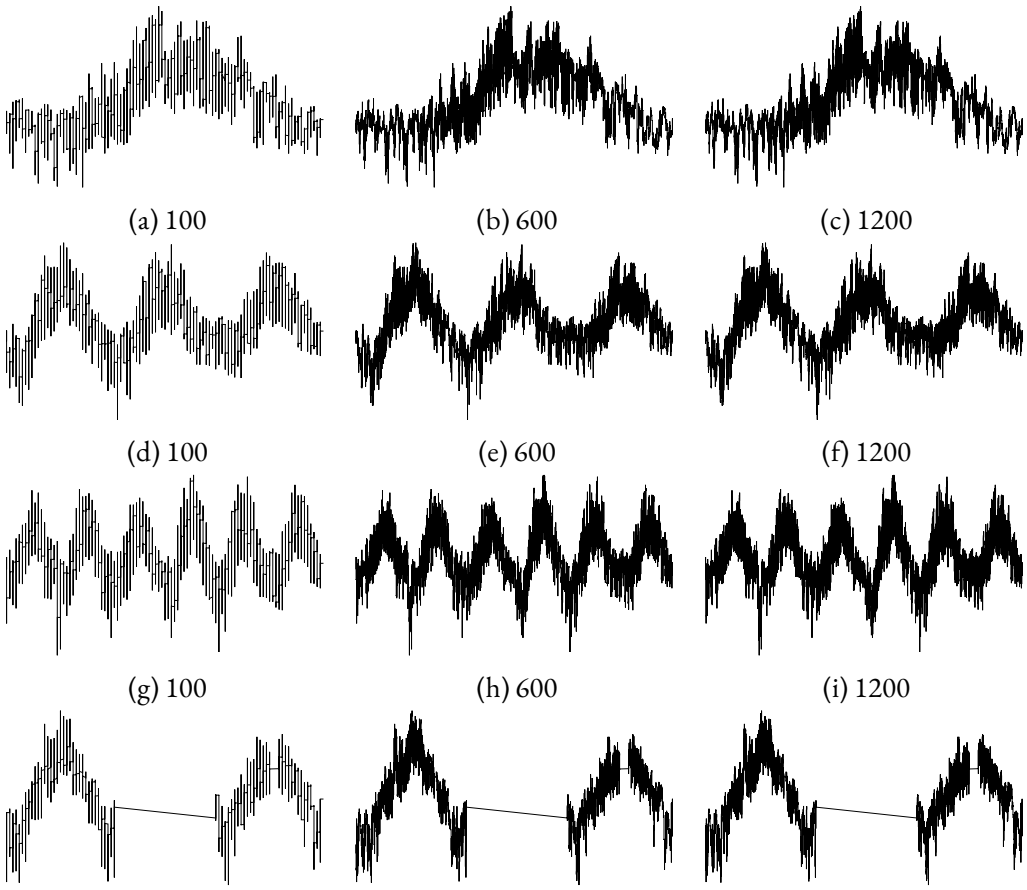
För M4 verkar bildkvaliteten hållas mer eller mindre konstant i alla fyra tidsintervall. Indexvärdena för SSIM och MSE ska sättas i kontext, hur väl en viss bild med ett visst indexvärde i praktiken fungerar beror förstås på användningsområdet. Linjediagrammen

Tabell 8: Resultat för M4

Intervall	Överföring[T1]	Uppritning[T2]	SVG-storlek[T3]	MSE[T4]	SSIM[T5]
2020	380 ms	8,6 ms	410 kB	860	0,96
2018–2020	1100 ms	9,1 ms	410 kB	770	0,97
2015–2020	2200 ms	9,3 ms	410 kB	910	0,96
2010–2012	380 ms	5,9 ms	270 kB	500	0,98

uppritade med M4 verkar ändå i detta arbete vara mycket nära referensbilderna, både enligt SSIM och MSE. Medelvärdet för SSIM som kan utläsas ur den sammanfattande tabellen 4, dvs. $SSIM(M4) = 0{,}97$, liknar resultat i Jugel m. fl. [25] där metoden uppvisat ett SSIM kring 0,95 för en upplösningsnivå på 2000. Det något högre värdet i detta arbete kommer av att medelvärdet sammanfattar också flera upplösningsnivåer högre än 2000.

Utöver de kvantitativa testerna kan konstateras att M4 också är en metod som både är enkel att tillämpa, om än en aning mer komplicerad än PAA, och fungerar väl i en webbapplikation med SQL-databas.



(a) 100 (b) 600 (c) 1200

(d) 100 (e) 600 (f) 1200

(g) 100 (h) 600 (i) 1200

(j) 100 (k) 600 (l) 1200

Figur 27: Det visuella resultatet i några upplösningsnivåer med M4

Genom att jämföra linjediagrammen i figur 27 med referensbilderna i figur 21 blir det klart att M4 leder till ett linjediagram som är mycket nära referensbilden. För att uppnå det resultatet verkar M4 kräva en upplösningsnivå som matchar det horisontella utrymmet i linjediagrammets koordinatsystem, upplösningsnivån 100 verkar förmedla den ursprungliga tidsseriens övergripande trend och periodicitet men ter sig glesare än referensbilden. Detta motsvarar också resultatet i Jugel m. fl. [25].

8.3 DFT

Likt de övriga metoderna leder också tillämpningen av metoden DFT till en snabbare överföring och uppritning av linjediagrammet. Överföringstiden hänger ändå starkt ihop med längden på den ursprungliga tidsserien och enligt tabell 9 är den i fallet 2015–2020 längre än de motsvarande resultaten för de tidigare metoderna PAA och M4.

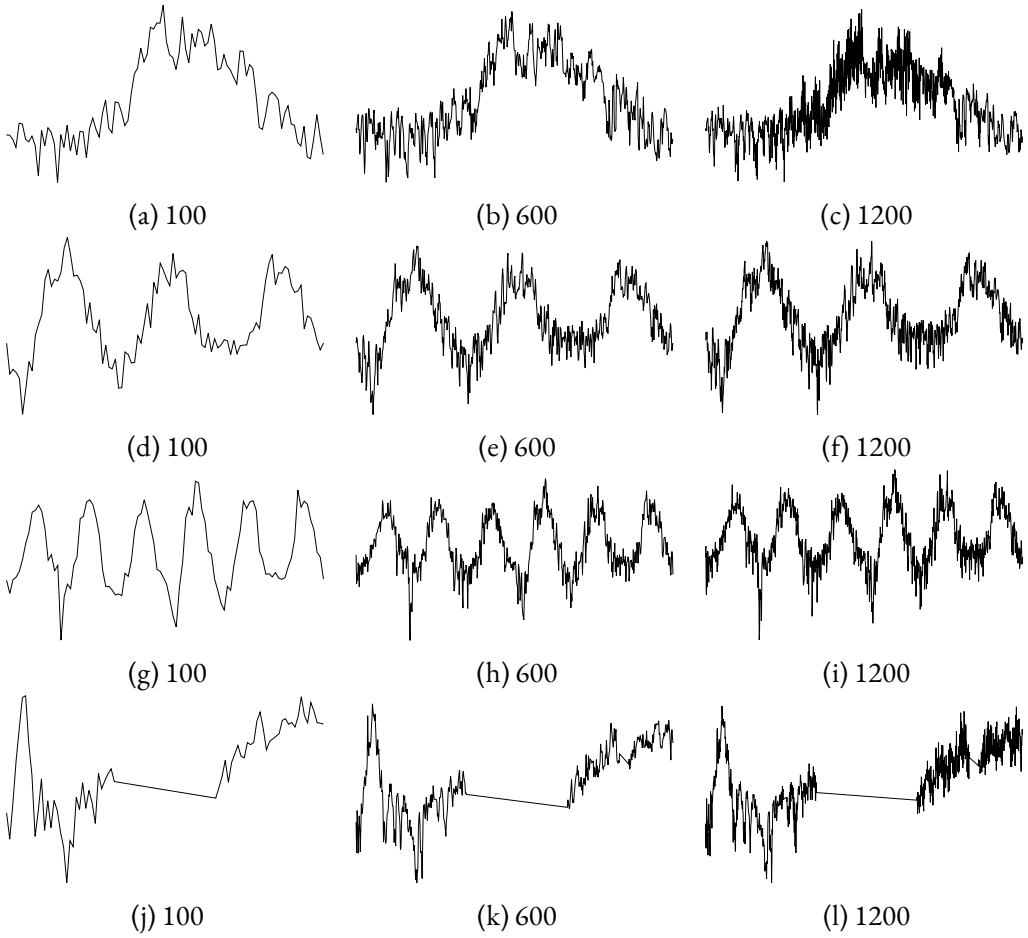
Tabell 9: Resultat för DFT

Intervall	Överföring[T1]	Uppritning[T2]	SVG-storlek[T3]	MSE[T4]	SSIM[T5]
2020	730 ms	3,4 ms	100 kB	5100	0,80
2018–2020	2200 ms	3,9 ms	110 kB	4800	0,82
2015–2020	4400 ms	7,9 ms	110 kB	5800	0,80
2010–2012	760 ms	2,7 ms	68 kB	6500	0,78

Enligt den sammanfattande tabellen 4 ger DFT de klart lägsta resultaten för bildkvalitet mätt i SSIM och MSE. Enligt resultaten från de enskilda intervallen i tabell 9 verkar det ändå som om metoden åstadkommer ett liknande SSIM-värde (0,82) för intervallet 2018–2020 som PAA i intervallet 2015–2020 (0,80). För DFT verkar bildkvaliteten inte direkt följa av hur lång den ursprungliga serien är, SSIM-värdet kretsar kring 0,8 för alla tre regelbundna intervall. I det oregelbundna intervallet 2010–2012 är SSIM-värdet lägre än i de andra fallen och t.ex. figur 28 (l) visar att resultatet för denna metod i det intervallet ordentligt avviker från referensbilden.

Metoden DFT tillämpas i detta arbete inte i databasen eftersom tillämpningar av den underliggande algoritmen inte finns tillhanda för SQL. För vanliga programmeringsspråk som Python och C++ finns det ändå färdiga implementationer av den underliggande algoritmen. Utöver de vanliga komponenterna som tillhör en webbapplikation kräver alltså den här metoden i regel ett skilt programbibliotek eller en skild komponent som står för att behandla tidsserien.

Fastän SSIM och MSE ger ett sämre resultat än de tidigare metoderna kan det ändå konstateras att linjediagrammen i figur 28 i de tre mera regelbundna intervallen helt subjektivt verkar ligga ganska nära referensbilderna. I den lägsta upplösningsnivån bibehålls



Figur 28: Det visuella resultatet i några upplösningsnivåer med DFT

trenden för data och i de högra nivåerna verkar de tätare partierna i tidsserien också urskiljas. För det oregelbundna fallet i intervall 2010–2012 är resultatet ändå långt ifrån referensbilden. Det går egentligen inte att känna igen den ursprungliga tidsserien i det med DFT reducerade linjediagrammet.

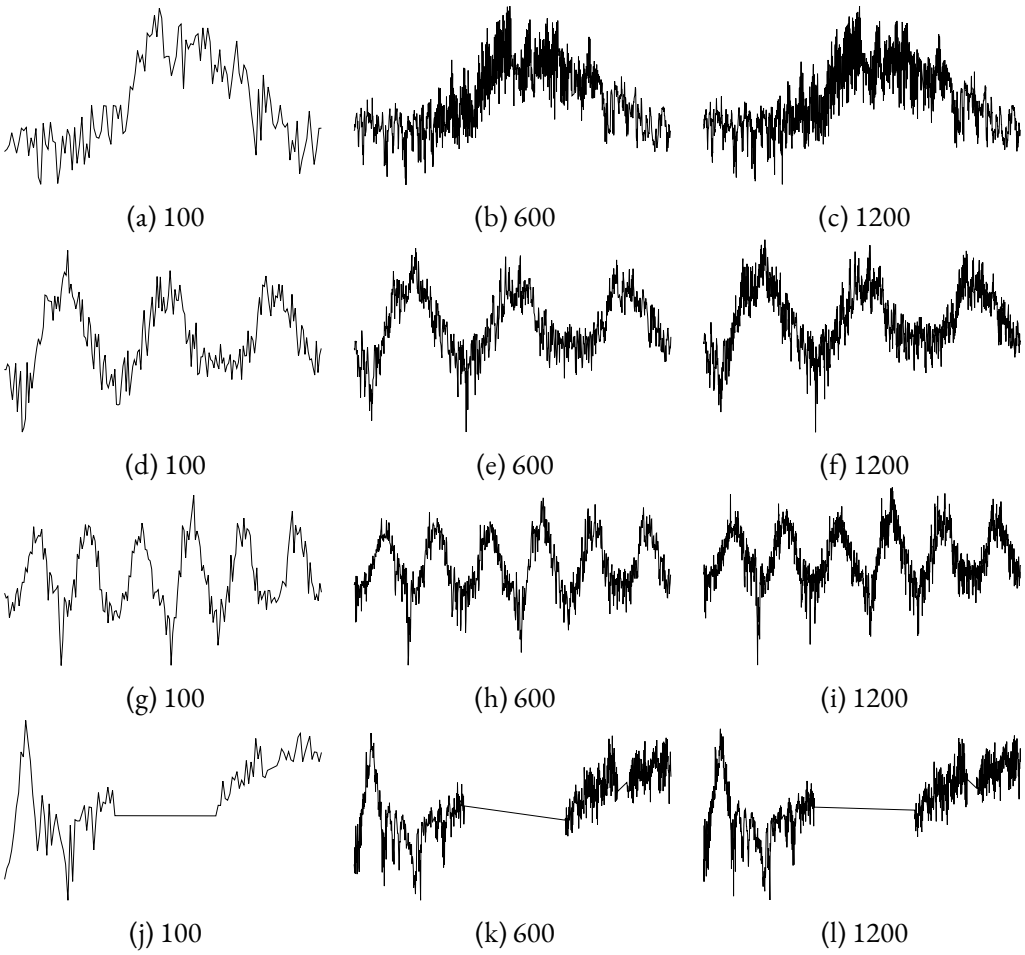
8.4 DWT

Enligt tabell 10 liknar resultatet för metoden DWT i stort resultatet för DFT som beskrevs i det föregående avsnittet. Bildkvaliteten i intervallen 2018–2020 och 2015–2020 är ändå högre för DWT än DFT enligt uppmätta värden för SSIM och MSE. På samma sätt som för DFT verkar överföringstiden för DWT öka i takt med att tidsintervallet och tidsserien blir längre. Metoden DWT uppvisar också fenomenet att bildkvaliteten inte direkt verkar hänga ihop med hur lång tidsserien är, till skillnad från metoden PAA där bildkvaliteten sjönk då intervallet blev längre verkar bildkvaliteten t.om. öka till en början för DWT då intervallet förändras. Metoden visar samma beteende som DFT i det oregelbundna intervallet 2010–2012 med ett lågt SSIM-värde och ett linjediagram i figur 29 (l) som är långt ifrån referensbilden. På samma sätt som DFT är också DWT

tillämpad i Python utanför databasen, och på samma sätt är metoden också relativt enkel att implementera med hjälp av färdiga bibliotek.

Tabell 10: Resultat för DWT

Intervall	Överföring[T1]	Uppritning[T2]	SVG-storlek[T3]	MSE[T4]	SSIM[T5]
2020	700 ms	4,8 ms	170 kB	4900	0,80
2018–2020	2000 ms	5,1 ms	150 kB	3800	0,85
2015–2020	4100 ms	5,4 ms	150 kB	4500	0,83
2010–2012	700 ms	4,4 ms	110 kB	6700	0,78



(a) 100 (b) 600 (c) 1200

(d) 100 (e) 600 (f) 1200

(g) 100 (h) 600 (i) 1200

(j) 100 (k) 600 (l) 1200

Figur 29: Det visuella resultatet i några upplösningsnivåer med DWT

9 Diskussion

Målet för detta arbete har varit att hitta en metod som på ett sparsamt sätt representerar den ursprungliga tidsserien, både gällande tiden som användaren väntar innan linjediagrammet är uppritat och gällande datamängden som servern producerar och webbläsaren bearbetar. Resultatet av testerna i det föregående kapitlet visar att alla fyra metoder som blivit behandlade kan tillämpas för att avlasta processen. Samtliga metoder kan användas för att skapa en representation som ger en inblick i den ursprungliga och obehandlade tidsserien och detta i rätt skala för att kunna ritas upp i samma koordinatsystem som originalet. Metoderna är också alla enkla att tillämpa eller så finns det fritt tillgängliga programpaket som implementerar metoderna på sätt som kunde fungera i praktiken på en webbsida.

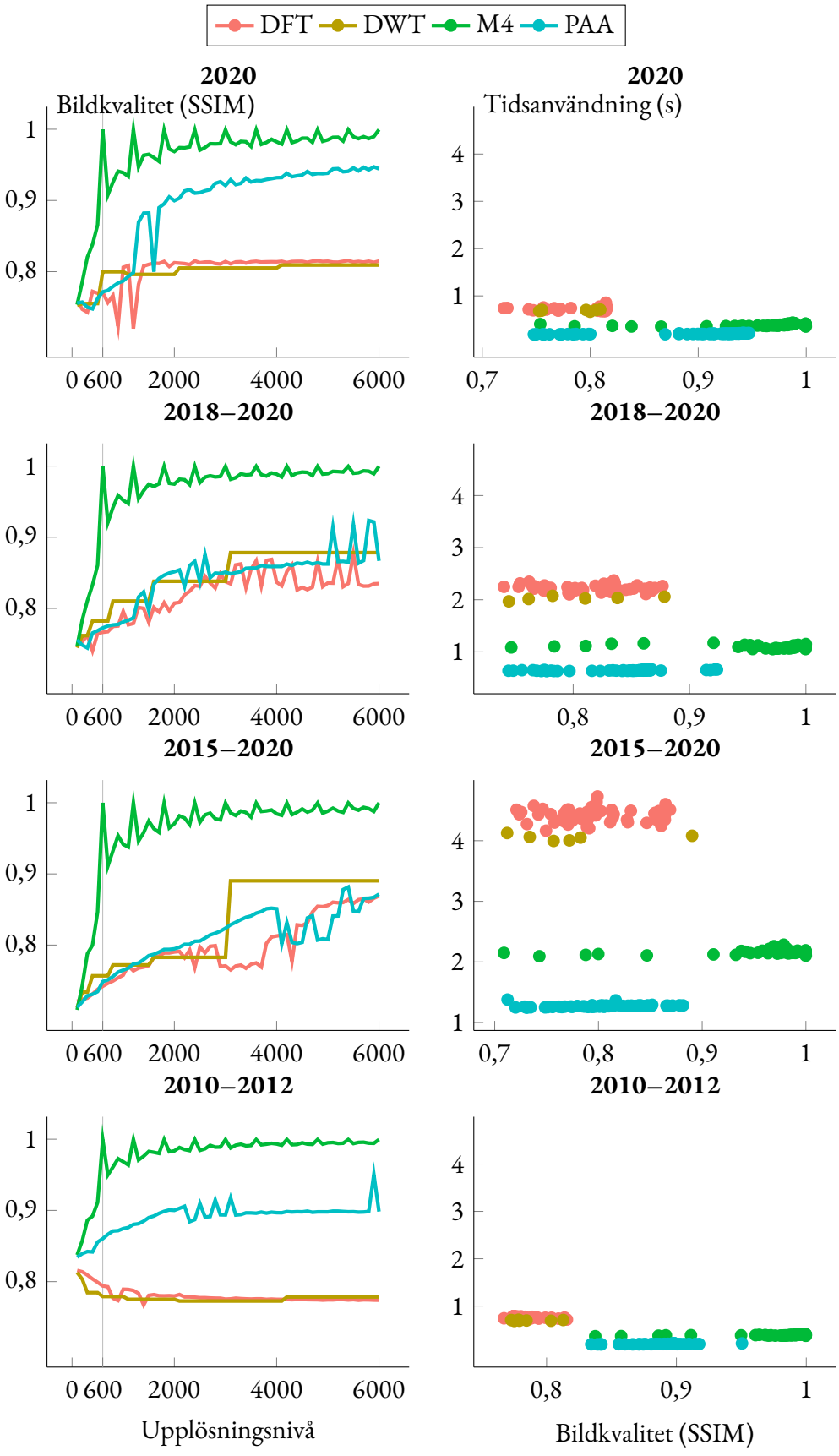
Enligt testresultaten i det föregående kapitlet liknar metoderna på många sätt varandra. Där de skiljer sig är hur nära metoderna lyckas återskapa den ursprungliga tidsseriens linjediagram och hur mycket snabbare än referensfallet de i praktiken åstadkommer datareduktionen. Testsystemet som presenterades lägger stor vikt vid en kvantitativ bildjämförelse med hjälp av SSIM-index och MSE-fel eftersom metoderna i flera fall rent subjektivt producerar diagram som liknar varandra. I praktiken beror ändå en jämförelse långt på de data som ska behandlas, de specifika kraven för tillämpningen och, med all sannolikhet, en helt subjektiv avvägning gällande vad som ser lämpligt ut.

Figur 30 sammanfattar resultaten från alla de 60 testkörningar som beskrevs i resultatkapitlet. Figuren består av två grupper av diagram, en grupp som visar hur det uppmätta SSIM-värdet, och därmed bildkvaliteten, förändras när upplösningsnivån höjs och en annan grupp som visar hur tidsanvändningen hänger ihop med bildkvaliteten.

9.1 Bildkvalitet

I allmänhet innebär fler detaljer en högre upplösning, och för bilder, speciellt fotografier, är fler detaljer ofta synonymt med fler pixlar. För metoderna i detta arbete verkar det ändå klart att fler mätpunkter inte automatiskt innebär en högre upplösning, dvs. att linjediagrammet innehåller fler detaljer ur det ursprungliga. Testresultatet verkar motivera en process där uttryckligen rätt, inte bara färre, mätpunkter ska väljas ut för att den ursprungliga tidsserien ska kunna representeras så bra som möjligt med den begränsade upplösningen som webbläsaren erbjuder.

Båda metoderna PAA och M4 använder den grundläggande idén att en förenklad version av den ursprungliga tidsserien kan skapas genom att dela in den i kortare intervall och för varje intervall välja något sätt att sammanfatta det. Metoden PAA förespråkar intervallens medelvärden som den sammanfattande biten information och i Keogh m. fl.



Figur 30: Sammanfattning av resultaten i upplösningsnivåerna 100–6000

[23] motiveras det med att det för naturliga data kan vara lämpligt pga. autokorrelationen som ofta påträffas i tidsserier som beskriver naturliga fenomen, dvs. att mätpunkter som ligger nära varandra i tiden också brukar anta liknande värden. Valet motiveras också av att medelvärdet är ett praktiskt värde att räkna ut. För metoden M4 väljs flera värden för varje intervall och i Jugel m. fl. [25] motiveras det med hur de vanliga rasteriseringsalgoritmerna fungerar då en linje ritas upp på en datorskärm.

I den första gruppen av diagram i figur 30 verkar det tydligt att M4 leder till en högre bildkvalitet än PAA oberoende av upplösningsnivån. I till exempel det relativt korta intervall som år 2020 utgör ger metoden M4 ett SSIM-värde som inte går under 0,95 i någon upplösningsnivå högre än 1300 medan PAA endast närmar sig denna bildkvalitet efter upplösningsnivån 5000.

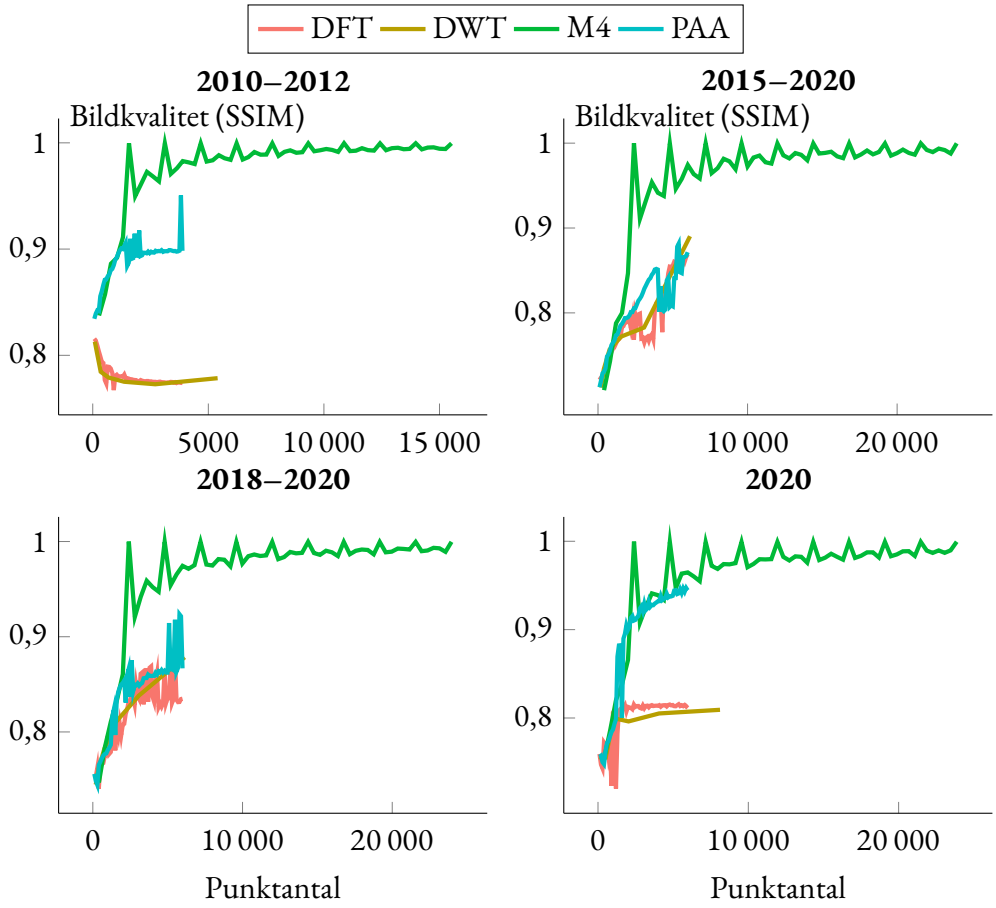
Upplösningsnivån 1300 i M4 motsvarar i praktiken en tidsserie på ungefär 5200 punkter, dvs. metoden PAA skulle producera detta antal punkter vid upplösningsnivån 5200. I figur 31 jämförs antalet punkter i den reducerade tidsserien med bildkvaliteten mätt i SSIM. För metoderna PAA och DFT är detta samma jämförelse som i figur 30 då punktantalet för dessa metoder motsvarar upplösningsnivån. I figur 31 är den vågräta axeln begränsad till 10 000 punkter då detta synliggör metodernas beteende bättre.

Den största skillnaden mellan de två figurerna är att figur 31 visar att metoden PAA klarar sig bättre än M4 vid lägre punktantal i de kortare intervallen år 2020 och åren 2010–2012. Detta förklaras möjligen av det mönster som syns i M4 vid upplösningsnivåer som är långt under den horisontella upplösningen, dvs. i det här fallet 600 punkter, jämför till exempel figurerna 26 och 27. Faktumet att M4 väljer fyra punkter verkar inte fungera lika bra då upplösningsnivån är för låg för det utrymme som linjediagrammet ritas upp i. Figur 31 illustrerar också beteendet för metoden DWT där punktantalet inte ökas konstant utan hoppar mellan de möjliga nivåerna enligt beskrivningen av tillämpningen i avsnitt 7.8.

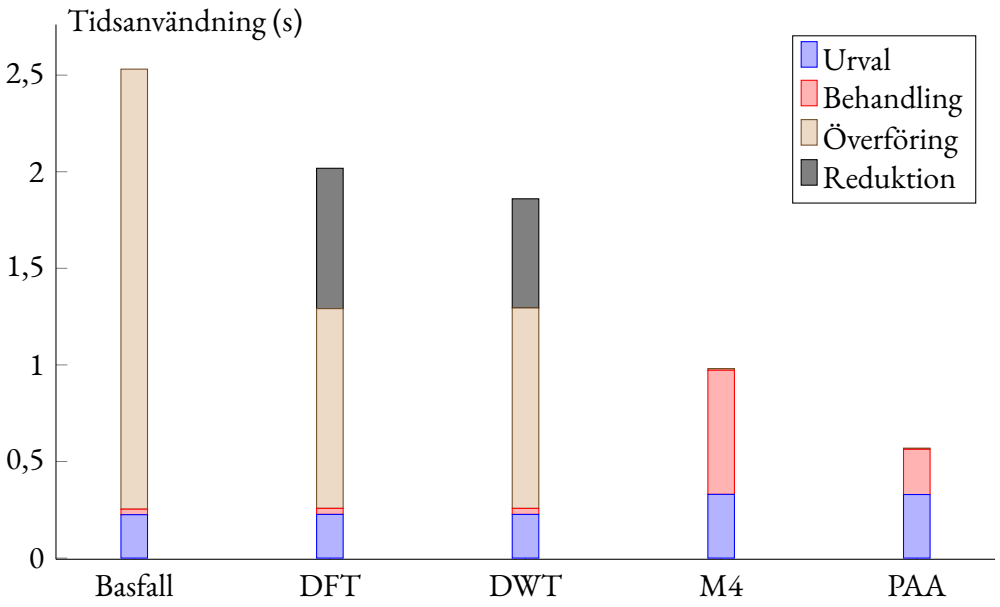
Metoden M4 verkar klart vara mer framgångsrik än PAA i de längre intervallen i figur 30. De uppmätta SSIM-värdena är nästan lika i alla tre testintervall medan resultatet för PAA är sämre ju längre intervallet är. Eftersom SSIM-indexet för M4 är så konstant i alla de fyra fallen och eftersom framgången för PAA i figur 31 inte i de längre intervallen når M4 verkar testkörningarna ge vid handen att det går att åstadkomma ett bättre resultat genom att tänka på hur diagrammet ritas upp i webbläsaren.

9.2 Tidsanvändning

En utmaning för en tillämpning som använder sig av metoderna DFT och DWT är att de i praktiken inte beräknas i databasen, utan de kräver båda att hela tidsserien hämtas ur databasen för att sedan behandlas vidare, ofta med någon programvara för numeriska be-



Figur 31: Bildkvaliteten jämfört med antalet punkter i den reducerade serien



Figur 32: Tidsanvändning i databasen och i Python-lagret

räkningar eller simulering. Figur 32 visar hur lång tid de huvudsakliga stegen i processen på servern tar i anspråk för de fyra metoderna. Figuren visar medeltalet för varje moment beräknat över de fyra testintervallen som beskrivs i avsnitt 7.2. I sammanfattningen ingår alltså inte den tid som krävs för att skicka data till klienten eller den tid som klienten tar i anspråk då diagrammet ritas upp. I figuren sker stegen *urval* och *behandling* i databasen och stegen *överföring* och *reduktion* på applikationsservern.

I urvalsskedet sker avgränsningen av den ursprungliga tidsserien till det tidsintervall som användaren valt. För basfallet och metoderna DFT och DWT innebär detta en SQL-förfrågning som bara väljer ut data med vissa kriterier, för metoderna M4 och PAA beräknar databasen i det här skedet också de tidsintervall som kommer att användas i sammanfattningen. Figuren visar också följaktligen att urvalet sker något långsammare för metoderna M4 och PAA.

I behandlingsskedet sammanfattas och sorteras data. För basfallet och metoderna DFT och DWT innebär detta endast att uthämtade data sorteras, för M4 och PAA är detta skedet som tillämpar den huvudsakliga datareduktionen. I figur 32 är detta skede också klart längre för de två senare metoderna eftersom databasen helt enkelt utför mera arbete. Här syns också att metoden PAA kräver mindre beräkningar än metoden M4.

Efter dessa två skeden överförs data från databasen till applikationsservern för vidare behandling. För basfallet och metoderna DFT och DWT innebär detta att en oreducerad tidsserie med alla mätpunkter inom det önskade tidsintervallet överförs från databasen till applikationsservern. För metoderna M4 och PAA krävs det bara att den betydligt kortare och redan reducerade tidsserien överförs, i figur 32 är detta steg för metoderna därmed praktiskt taget osynligt. Överföringen omfattar flera steg där bl.a. data först omvandlas till det format som databasen föreskriver för att kommuniceras över databasens protokoll till applikationsservern varefter applikationsservern också får göra en omvandling genom att anpassa datan till det format som krävs för att behandla dem vidare.

Denna överförings- och omvandlingsprocess är den främsta orsaken till att de båda signalbehandlingsmetoderna fungerar långsammare än M4 och PAA i detta arbete. Det verkar motiverat att från databasen bara överföra så lite data som möjligt.

I figur 30 ingår förutom diagrammen över hur bildkvaliteten förändras i takt med upplösningsnivån också diagram som beskriver metodernas tidsanvändning som en funktion av resultaten i bildkvalitetstestet. Det figuren visar är att tidsanvändningen ökar då det ursprungliga intervallet blir längre. Intressant är att se att tidsanvändningen inte nämnvärt påverkas av det uppnådda SSIM-värdet, för varje metod och intervall verkar testkörningarna hålla sig kring samma tidsanvändning. Detta resultat verkar motivera att ett högt SSIM-värde, dvs. en nära avbild av det ursprungliga diagrammet, inte ska ses som den största orsaken till en hög tidsanvändning.

10 Slutord

För att visa hur ett fenomen utvecklar sig över tid är ofta linjediagrammet det rätta verktyget, för i ett linjediagram fästs inte uppmärksamheten vid de enskilda mätpunkterna utan trenden överlag är vad läsaren uppmuntras att studera. Genom att lämna bort sådana punkter ur en tidsserie som inte avsevärt påverkar utseendet kan ett linjediagram ritas upp på ett mer effektivt sätt utan att påverka diagrammets ändamål.

I detta arbete låg vikten vid det tidseffektiva och det visuella. Problemet som skulle lösas var att överföringen och uppritningen av kompletta tidsserier i samband med en webbsida ofta helt enkelt är för långsamma. Genom att välja bort mätpunkter kunde den ursprungliga tidsserien förenklas och hur denna förenkling påverkade utseendet av linjediagrammet diskuterades.

Det verkar klart att det finns flera sätt att närma sig problemet. Om målet är att ett förändrat linjediagrams utseende ska vara så likt det ursprungliga som möjligt, verkar det ändå vara klart att utgångspunkten för en metod också måste ligga i det visuella.

Då data ska kommuniceras är målet förstås i praktiken att meddelandet förmedlas till läsaren och för det krävs det att en datavisualisering är gjord med eftertanke. Hurdant idealet, som linjediagrammet ska efterlikna, ser ut är något som detta arbete inte tog ställning till eftersom det beror på användningsområdet. De fyra metoderna som presenterades i arbetet gav olika slutresultat men hur väl de i praktiken fungerar beror mycket på de data och de tankar som ska kommuniceras.

De uppmätta SSIM-värdena i resultatkapitlet, som beskriver hur nära de ursprungliga linjediagrammen avbildats, ska inte uppfattas som verkliga egenskaper för metoderna utan de uppkom som funktioner av metoderna, de data som användes och sättet som materialet presenterades i testerna.

För metoden M4 blev det ändå klart att det fenomen som redan diskuterats i litteraturen också gick igen i denna studie, dvs. att kunskap om rasteriseringsalgoritmen kan utnyttjas för att skapa en snarlik avbild av det ursprungliga linjediagrammet. Metoden hade det högsta SSIM-värdet i alla upplösningsnivåer och digrammen som producerades var mycket lika referensbilderna. En tillämpning av metoden kunde t.o.m. ersätta uppritningen av det ursprungliga materialet utan att förleda slutanvändaren.

Till skillnad från behandlingen i Jugel m. fl. [25], där linjediagram uttryckligen ritades upp som pixelbaserade bilder, ritades diagrammen här upp med hjälp av vektorgrafik. Resultatet blev ändå liknande på grund av den kontroll som webbläsaren ger webbutvecklaren att bestämma hur illustrationer visas för slutanvändaren, dvs. med dimensioner som följer pixelgittret i skärmen och avstängd kantutjämning. I Jugel [44] behandlas hur metoden kunde anpassas för att också tillämpas på diagram där kantutjämningen är viktig men detta arbete tog inte ställning till den aspekten.

Metoden PAA beskrevs i litteraturen som ett sätt att förenkla tidsserier som sedan sparas i en databas och används för dataurvinning. Tillämpningen av metoden i detta arbete var alltså delvis ryckt ur sin kontext, men eftersom denna metod uttrycker en ganska enkel idé om hur tidsserier kan förenklas och trender illustreras verkade detta ändå motiverat. Med hjälp av PAA kunde förenklade diagram ritas upp som förmedlade hur det ursprungliga diagrammet såg ut överlag. Till skillnad från metoden M4 verkar det ändå klart att denna metod ger ett diagram som ändå avviker rätt mycket från det ursprungliga, speciellt i de lägre upplösningsnivåerna. Eftersom metoden är så enkel att tillämpa och snabb att beräkna verkar den ändå användbar, men en tillämpning av metoden enligt det mönster som beskrivs i detta arbete torde kräva att webbutvecklaren gör det klart för slutanvändaren att det som visas bara är en approximation av den ursprungliga tidsserien.

Metoden DFT med sitt ursprung i frekvensanalys var intressant att studera eftersom den inte behandlar signalen bitvis utan representationen tar i beaktande hela serien. Överlag var resultatet inte att jämföra med resultatet för metoderna M4 och PAA, men så som det också presenterats i litteraturen verkade uttrycksförmågan för ett begränsat antal koefficienter vara bättre i de mer periodiska testfallen 2018–2020 och 2015–2020. Tillämpningen av denna metod krävde fler komponenter på servern än bara den sedvanliga databasen och applikationsservern, det krävdes programvara för numeriska beräkningar. Här ligger också akilleshälen för denna typ av lösning. Eftersom serierna måste överföras i sin helhet från databasen till applikationsservern sker ett tidssvinn på servern som är svårt att ta igen. Det verkar svårt att rekommendera denna typ av lösning om den inte åstadkommer en avsevärt bättre visualisering för att kompensera den längre tiden som krävs för behandling på servern.

Metoden DWT är intressant på grund av hur den behandlar upplösningsnivåer och tidslokalitet. Olika intervall av en signal kan med hjälp av metoden representeras i olika upplösningar på samma gång. Detta studerades ändå inte i detta arbete utan lösningen som presenterades var mycket enkel och resultatet blev liknande som för DFT.

För båda metoderna DFT och DWT finns det flera sätt som tillämpningarna kunde utvecklas. För metoden DFT kunde det t.ex. vara intressant att studera om en tvådimensionell transform kunde skapa ett diagram som närmare efterliknar det ursprungliga. Med en liknande process som i bildformatet JPEG kunde tidsserien behandlas i två dimensioner genom att dela in koordinatsystemet i block av någon bestämd storlek för vilka sedan en transform beräknas. De block genom vilka ingen linje dras kunde sedan komprimeras bort.

För metoden DWT kunde dess stöd för en progressiv komprimering studeras, i stil med hur den används i JPEG2000. För en tidsserie som kanske inte behöver visas interaktivt, utan kan förenklas på förhand, kan en DWT beräknas i flera nivåer och sparas i databasen.

När visualiseringen sedan ska skapas i webbläsaren kunde data skickas progressivt, en nivå i taget, tills en önskad upplösning nåtts. På det sättet kunde slutanvändaren direkt få respons från servern och de högre upplösningarna kunde hämtas in i bakgrunden.

Metoderna PAA och M4 klarade sig som sagt relativt bra i denna jämförelse. För PAA kunde det ändå vara intressant att utveckla tankegången och prova på hur andra lägesmått än medelvärdet påverkar utseendet på diagrammet, t.ex. medianen. För M4 kunde det vara intressant att undersöka hur variansen i data bättre kunde uttryckas. Med metoden blir resultatet för bilder med låg upplösning lätt stora ytor som är helt svarta och då finns risken att information om hur många punkter varje intervall faktiskt innehöll kan falla bort. För detta kunde det t.ex. vara en möjlighet att utnyttja en gråskala för att förmedla mer information om varje intervall.

Utöver metoderna som presenterades i detta arbete finns det också en uppsjö andra metoder för komprimering och behandling av tidsserier som inte alls behandlats i detta arbete. Om utgångspunkten är det visuella och det viktiga är att en förenkling är lämplig rent visuellt sett, kan t.ex. metoder för förenkling av linjer vara intressant. Inom kartritning har algoritmer som Ramer-Douglas-Peucker (Ramer [45] och Douglas och Peucker [46]) och nyare som Visvalingam-Whyatt (Visvalingam och Whyatt [47]) presenterats som metoder för att förenkla linjer genom att ta bort de partier som verkar ointressanta. Om utgångspunkten däremot är att metoden ska vara tidseffektiv, kan det vara intressant att studera någon mer avancerad form av komprimering, t.ex. den prediktiva komprimeringen som beskrivs i Blalock m. fl. [48].

Källförteckning

[1] C. Cavallin, *Swensk-Latinsk Ordbok*, Digital utgåva (Projekt Runeberg). Lund: F. & G. Beijers förlag, 1875.

[2] H.-J. Rheinberger, ”Infra-experimentality: From traces to data, from data to patterning facts”, *History of Science*, årg. 49, nr 3, s. 337–348, 2011.

[3] H. Kennedy m. fl., ”The work that visualisation conventions do”, *Information, Communication & Society*, årg. 19, nr 6, s. 715–735, 2016.

[4] D. Beer och R. Burrows, ”Popular Culture, Digital Archives and the New Social Life of Data”, *Theory, Culture & Society*, årg. 30, nr 4, s. 47–71, 2013. DOI: 10.1177/0263276413476542. URL: https://doi.org/10.1177/0263276413476542.

[5] M. Bhuller m. fl., ”How the internet changed the market for print media”, tekn. rapport, 2020.

[6] N. Sung och J. Kim, ”Does the internet kill newspapers? The case of South Korea”, *Telecommunications Policy*, årg. 44, nr 4, s. 101 955, 2020. DOI: https://doi.org/10.1016/j.telpol.2020.101955. URL: https://www.sciencedirect.com/science/article/pii/S0308596120300471.

[7] J. Jenkins och R. K. Nielsen, ”The digital transition of local news”, 2018.

[8] E. R. Tufte, *The Visual Display of Quantitative Information*, 2. utg. Cheshire, CT, USA: Graphics Press, 2001.

[9] M. Friendly, ”A Brief History of Data Visualization”, i *Handbook of Data Visualization*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, s. 15–56. DOI: 10.1007/978-3-540-33037-0_2. URL: https://doi.org/10.1007/978-3-540-33037-0_2.

[10] A. Taivalsaari m. fl., ”Comparing the Built-In Application Architecture Models in the Web Browser”, i *2017 IEEE International Conference on Software Architecture (ICSA)*, 2017, s. 51–54. DOI: 10.1109/ICSA.2017.23.

[11] K. Nygård, ”Single page architecture as basis for web applications”, examensarb., Aalto University, 6 juni 2015.

[12] NIST/SEMATECH, *e-Handbook of Statistical Methods*, 2012. DOI: 10.18434/M32189. (hämtad 2021-06-15).

[13] J. W. Tukey, *Exploratory Data Analysis*. Reading, MA, USA: Addison-Wesley, 1977.

[14] J. Koponen och J. Hildén, *Data visualization handbook*. Aalto ARTS Books, Esbo, 2019.

[15] J. Schwabish, *Better Data Visualizations: A Guide for Scholars, Researchers, and Wonks*. Columbia University Press, 2021. DOI: 10.7312/schw19310.

[16] P. Esling och C. Agon, ”Time-series data mining”, *ACM Computing Surveys (CSUR)*, årg. 45, nr 1, s. 1–34, 2012.

[17] H. Madsen, *Time series analysis*. Boca Raton, FL, USA: CRC Press, 2007.

[18] C. Aggarwal, *Data mining: the textbook*. Springer, 2015.

[19] Meteorologiska Institutet, *Väderobservationer – Lufttemperatur vid Åbo flygplats fr.o.m. 1.1.1960 t.o.m. 31.12.2021 [CC BY 4.0]*, Meteorologiska institutet, 2021. URL: https://www.ilmatieteenlaitos.fi/havaintojen-lataus.

[20] L. Tan och J. Jiang, *Digital signal processing*, 3. utg. Academic Press, 2019. DOI: 10.1016/C2017-0-02319-4.

[21] R. Agrawal m. fl., ”Efficient similarity search in sequence databases”, i *Foundations of Data Organization and Algorithms*, D. B. Lomet, utg., Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, s. 69–84.

[22] K.-P. Chan och A. W.-C. Fu, ”Efficient time series matching by wavelets”, i *Proceedings 15th International Conference on Data Engineering (Cat. No.99CB36337)*, 1999, s. 126–133. DOI: 10.1109/ICDE.1999.754915.

[23] E. Keogh m. fl., ”Dimensionality reduction for fast similarity search in large time series databases”, *Knowledge and information Systems*, årg. 3, nr 3, s. 263–286, 2001.

[24] J. Lin m. fl., ”A Symbolic Representation of Time Series, with Implications for Streaming Algorithms”, i *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, ser. DMKD ’03, San Diego, California: Association for Computing Machinery, 2003, s. 2–11. DOI: 10.1145/882082.882086.

[25] U. Jugel m. fl., ”M4: A Visualization-Oriented Time Series Data Aggregation”, *Proc. VLDB Endow.*, årg. 7, nr 10, s. 797–808, 2014. DOI: 10.14778/2732951.2732953.

[26] D. Mazzoni och R. B. Dannenberg, ”A Fast Data Structure for Disk-Based Audio Editing”, *Computer Music Journal*, årg. 26, nr 2, s. 62–76, 2002. DOI: 10.1162/014892602760137185.

[27] A. C. Bovik, ”1.1 - Introduction to Digital Image and Video Processing”, i *Handbook of Image and Video Processing (Second Edition)*, ser. Communications, Networking and Multimedia, A. BOVIK, utg., Second Edition, Burlington: Academic Press, 2005, s. 3–I. DOI: https://doi.org/10.1016/B978-012119792-6/50065-6. URL: https://www.sciencedirect.com/science/article/pii/B9780121197926500656.

[28] Z. Wang och A. C. Bovik, ”Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures”, *IEEE Signal Processing Magazine*, årg. 26, nr 1, s. 98–117, 2009. DOI: 10.1109/MSP.2008.930649.

[29] Z. Wang m. fl., ”Image quality assessment: from error visibility to structural similarity”, *IEEE Transactions on Image Processing*, årg. 13, nr 4, s. 600–612, 2004. DOI: 10.1109/TIP.2003.819861.

[30] S. van der Walt m. fl., ”scikit-image: image processing in Python”, *PeerJ*, årg. 2, e453, juni 2014. DOI: 10.7717/peerj.453. URL: https://doi.org/10.7717/peerj.453.

[31] I. Jacobs och N. Walsh, ”Architecture of the World Wide Web, Volume One”, W3C, W3C Recommendation, dec. 2004, https://www.w3.org/TR/2004/REC-webarch-20041215/.

[32] A. Grosskurth och M. Godfrey, ”A reference architecture for Web browsers”, i *21st IEEE International Conference on Software Maintenance (ICSM'05)*, 2005, s. 661–664. DOI: 10.1109/ICSM.2005.13.

[33] W3Techs. ”Usage statistics of client-side programming languages for websites”. (27 okt. 2021), URL: https://w3techs.com/technologies/overview/client_side_language.

[34] S.-y. Guo m. fl., ”ECMAScript 2022 Language Specification”, Ecma International, tekn. rapport, 23 okt. 2021. URL: https://tc39.es/ecma262.

[35] L. D. Paulson, ”Building rich web applications with Ajax”, *Computer*, årg. 38, nr 10, s. 14–17, 2005.

[36] A. Bellamy-Royds m. fl., *Using SVG with CSS3 and HTML5*, M. Foley, utg. CA, USA: O'Reilly Media, Inc., 26 april 2019.

[37] T. Bray, *The JavaScript Object Notation (JSON) Data Interchange Format*, RFC 7159, mars 2014. DOI: 10.17487/RFC7159. URL: https://www.rfc-editor.org/info/rfc7159.

[38] npm, *About npm*, 27 okt. 2021. URL: https://docs.npmjs.com/about-npm.

[39] J. S. Yi m. fl., ”Toward a deeper understanding of the role of interaction in information visualization”, *IEEE transactions on visualization and computer graphics*, årg. 13, nr 6, s. 1224–1231, 2007.

[40] A. Bouch m. fl., ”Quality is in the eye of the beholder: Meeting users’ requirements for internet quality of service”, i *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2000, s. 297–304.

[41] J. Nielsen. ”Website response times”. (2010), URL: https://www.nngroup.com/articles/website-response-times/.

[42] C. R. Harris m. fl., ”Array programming with NumPy”, *Nature*, årg. 585, nr 7825, s. 357–362, sept. 2020. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[43] G. R. Lee m. fl., *PyWavelets/pywt: PyWavelets 1.2.0*, version v1.2.0, nov. 2021. DOI: 10.5281/zenodo.5655782. URL: https://doi.org/10.5281/zenodo.5655782.

[44] U. Jugel, ”Visualization-driven data aggregation : rethinking data acquisition for data visualizations”, avhandling, Technische Universität Berlin, Berlin, 2017. DOI: 10.14279/depositonce-6561.

[45] U. Ramer, ”An iterative procedure for the polygonal approximation of plane curves”, *Computer graphics and image processing*, årg. 1, nr 3, s. 244–256, 1972.

[46] D. H. Douglas och T. K. Peucker, ”Algorithms for the reduction of the number of points required to represent a digitized line or its caricature”, *Cartographica: the international journal for geographic information and geovisualization*, årg. 10, nr 2, s. 112–122, 1973.

[47] M. Visvalingam och J. D. Whyatt, ”Line generalisation by repeated elimination of points”, *The cartographic journal*, årg. 30, nr 1, s. 46–51, 1993.

[48] D. Blalock m. fl., ”Sprintz: Time series compression for the internet of things”, *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, årg. 2, nr 3, s. 1–23, 2018.

Bilagor

```sql
WITH Q AS (
  SELECT
    date_bin ((:t_end - :t_start) / :points, "timestamp", :t_start),
    m.temperature
  FROM
    measurements AS m
  WHERE
    "timestamp" BETWEEN :t_start AND :t_end
)
SELECT
  date_bin AS "timestamp",
  avg(temperature) AS temperature
FROM
  Q
GROUP BY
  "timestamp"
ORDER BY
  "timestamp";
```

Kodlistning 1: SQL-förfrågning som tillämpar PAA-metoden för datareduktion

```
WITH Q AS (
  SELECT
    date_bin ((:t_end - :t_start) / :points, "timestamp", :t_start),
    "timestamp",
    m.temperature
  FROM
    measurements AS m
  WHERE
    "timestamp" BETWEEN :t_start AND :t_end
),
agg AS (
  SELECT
    q.date_bin,
    array_agg(temperature ORDER BY "timestamp") AS arr
FROM
  Q
GROUP BY
  date_bin
ORDER BY
  date_bin
)
SELECT
  date_bin AS "timestamp",
  arr[1] AS min_t,
  (
    SELECT
      max(i)
    FROM
      unnest(arr) i) max_v,
  (
    SELECT
      min(i)
    FROM
      unnest(arr) i) min_v,
  arr[array_length(arr, 1)] max_t
FROM
  agg
ORDER BY
  "timestamp";
```

Kodlistning 2: SQL-förfrågning som tillämpar M4-metoden för datareduktion