

Teknisk dokumentation

Redaktör: Anders Rehult

Version xx

Status

| | | |
|----------|-----|----------|
| Granskad | DOK | 18-03-05 |
| Godkänd | | |

PROJEKTIDENTITET

2018/VT, Grupp 2
Linköpings Tekniska Högskola, IFM

Gruppdeltagare

| Namn | Ansvar | Telefon | E-post |
|------------------|------------------------|-------------|-------------------------|
| Anders Rehult | Projektledare (PL) | 076-3161206 | andre449@student.liu.se |
| Marian Brännvall | Dokumentansvarig (DOK) | 070-7280044 | marbr639@student.liu.se |
| Andreas Kempe | Sekreterare (SE) | 073-9796689 | andke133@student.liu.se |
| Viktor Bernholtz | Viktor Bernholtz (VB) | 073-0386030 | vikbe253@student.liu.se |

Kund: IFM, Linköpings universitet, 581 83 Linköping

Kontaktperson hos kund: Rickard Armiento, 013-281249, rickard.armiento@liu.se

Kursansvarig: Per Sandström, 013-282902, persa@ifm.liu.se

Handledare: Johan Jönsson, 013-281176, johan.jonsson@liu.se

Innehåll

| | |
|---|-----------|
| Dokumenthistorik | v |
| 1 Inledning | 1 |
| 1.1 Parter | 1 |
| 1.2 Syfte och mål | 1 |
| 1.3 Användning | 1 |
| 1.4 Begränsningar | 1 |
| 1.5 Definitioner | 2 |
| 2 Översikt av systemet | 3 |
| 2.1 Resurser | 3 |
| 2.2 Ingående delsystem | 3 |
| 2.3 Utvecklingsfilosofi | 3 |
| 3 Delsystem 1 - System för parsning | 4 |
| 3.1 VASP | 4 |
| 3.2 HDF5-struktur för VASP-data | 4 |
| 3.3 Modul för skrivning till HDF5 | 5 |
| 3.4 Moduler för parsning | 7 |
| 3.4.1 Incarparser | 8 |
| 3.4.2 Volymparser | 8 |
| 3.4.3 Tillståndstäthetsparser | 9 |
| 3.4.4 Fermi-yteparser | 9 |
| 3.4.5 Enhetscellsparser | 9 |
| 3.4.6 parse_all | 10 |
| 4 Delsystem 2 - System för visualisering | 10 |
| 4.1 Nätverk | 10 |
| 4.1.1 Nätverk för visualisering av volymdata | 11 |
| 4.1.2 Nätverk för visualisering av enhetscell | 15 |
| 4.1.3 Nätverk för visualisering av DOS | 17 |
| 4.2 Datastrukturer | 17 |
| 4.2.1 Point | 17 |
| 4.2.2 Function | 17 |
| 4.3 Processorer | 17 |
| 4.3.1 Kristallstruktur | 17 |

| | | |
|----------|---|-----------|
| 4.3.2 | HDF5 | 19 |
| 4.3.3 | 2D | 21 |
| 4.4 | Properties och widgets | 22 |
| 4.4.1 | IntVectorpropety | 22 |
| 4.4.2 | IntVectorPropertyWidget | 22 |
| 5 | Visualisering | 23 |
| 5.1 | Utritning | 23 |
| 5.2 | Användarindata | 24 |
| 5.3 | Interaktivitet | 25 |
| 5.4 | HDF5-indata | 25 |
| 5.5 | Kristallstruktur | 25 |
| 5.5.1 | Översiktlig beskrivning | 25 |
| 5.5.2 | Inwiwo-nätverk för visualisering av kristallstruktur | 25 |
| 5.6 | Elektrontäthet | 26 |
| 5.6.1 | Översiktlig beskrivning | 26 |
| 5.6.2 | Inwiwo-nätverk för visualisering av elektrontäthet | 26 |
| 5.7 | Tillståndstäthet | 27 |
| 5.7.1 | Översiktlig beskrivning | 27 |
| 5.7.2 | Inwiwo-nätverk för visualisering av tillståndstäthet | 27 |
| 5.8 | Fermi-ytor | 29 |
| 5.8.1 | Inwiwo-nätverk för Brillouin-zonuppritning | 29 |
| 5.8.2 | Inwiwo-nätverk för Fermi-yte-uppritning | 30 |
| 5.9 | Dynamisk visualiseing av kristallstruktur | 31 |
| 5.9.1 | Översiktlig beskrivning | 31 |
| 5.9.2 | Inwiwo-nätverk för dyanmisk visualisering av kristallstruktur | 31 |
| | Referenser | 33 |
| | Bilaga A HDF5-datastruktur | 34 |
| | Bilaga B HDF5-datastruktur i två delar | 35 |

Dokumenthistorik

| Version | Datum | Utförda förändringar | Utförda av | Granskad |
|---------|------------|----------------------|----------------|----------|
| 0.1 | 2018-02-27 | Första utkast. | Projektgruppen | SE |
| 0.2 | 2018-03-05 | Andra utkast. | Projektgruppen | DOK |

1 Inledning

Dokumentet är en teknisk dokumentation för kandidatprojektet i visualisering av elektronstrukturer. Visualisering av elektronstrukturer är ett av projekten i kursen TFYA75 vid Linköpings universitet. Den tekniska dokumentationen beskriver detaljerat hur systemet är implementerat.

1.1 Parter

Rickard Armiento har beställt systemet som är beskrivet i denna tekniska dokumentation. Medlemmarna i projektgruppen, som är listade under rubriken projektidentitet ovan, är mottagare av denna beställning och har haft i uppgift att implementera systemet. Projektgruppens handledare är Johan Jönsson.

1.2 Syfte och mål

Inom materialfysik är elektronstrukturberäkningar ett viktigt teoretiskt verktyg. Detta för att få en så bra förståelse som möjligt av hur olika material är uppbyggda, bland annat vad gäller kristallers egenskaper sett ur ett kvantmekaniskt perspektiv. Det kan exempelvis handla om att man vill ta reda på olika materials egenskaper vad gäller värmeledningsförmåga, strömledningsförmåga etc.

För att öka förståelsen samt för att enklare kunna presentera resultat från forskning inom området är det viktigt att kunna göra olika typer av visualiseringar baserat på elektronstrukturberäkningar.

I många av de system som används för att utföra dessa beräkningar, exempelvis VASP, ges ingen eller begränsad möjlighet till detta. Vidare är de system som idag finns tillgängliga för visualisering förhållandevis ineffektiva, dåligt standardiserade samt har få visualiseringsfunktioner.

Syftet och målet för projektet har varit att uppdatera och utöka det modifierbara verktyg för visualisering av resultaten från elektronstrukturberäkningar som togs fram av 2017 års projektgrupp. Utöver det konkreta målet med utvecklingen av mjukvara för visualisering ska även projektet ge projektmedlemmarna erfarenhet av att arbeta i projekt och utöka deras förmåga till analytiskt och fysikaliskt tänkande för att ge värdefull erfarenhet inför arbetslivet.

1.3 Användning

Denna produkt kommer användas vid Linköpings universitet för att analysera data från elektronstruktursberäkningar.

1.4 Begränsningar

I projektet kommer visualiseringsverktyget Inwiwo och programmeringspråken Python och C++ användas. Det kommer inte utredas om det är bättre att använda andra verktyg.

1.5 Definitioner

- **Inviwo:** Ett forskningsverktyg som utvecklas vid Linköpings universitet och ger användaren möjlighet att styra visualisering med hjälp av programmering i Python 3 eller grafiskt. Det tillhandahåller även användargränssnitt för interaktiv visualisering.
- **Processor:** Benämningen på ett funktionsblock i Inviwos nätverksredigerare som tar emot indata och producerar utdata. I detta dokument avser en processor alltid en inviwoprocessor om inte annat anges.
- **Ports:** Kanaler som processorer använder för att utbyta data av specifika typer.
- **MultiInports:** Ports som kan ta emot flera datastrukturer av samma typ.
- **Properties:** Variabler som bestämmer en processors tillstånd.
- **Länkar:** Kanaler som processorer använder för att länka samman properties av samma typ så att deras tillstånd synkroniseras.
- **Inviwo-nätverk:** Ett antal processorer sammankopplade via portar och länkar.
- **vec3:** Vektor med tre komponenter.
- **Mesh:** Polygonyta.
- **Volymdata:** Tredimensionella data.
- **API** (Application Programming Interface): En specifikation av hur olika applikationer kan använda och kommunicera med en specifik programvara. Detta utgörs oftast av ett dynamiskt länkat bibliotek. [9]
- **BSD2** är en licens för öppen källkod. [3]
- **C++** är ett programmeringsspråk. [4]
I Inviwo används C++ för att skriva programkod till processorer.
- **Python** är ett programmeringsspråk. [14]
I Inviwo används Python för att knyta samman processorer.
- **Fermi-energi** definieras som en energinivå där antalet tillstånd som har en energi lägre än Fermi-energin är lika med antalet elektroner i systemet. [2]
- **Fermi-yta** är, för elektroners k -punkter i reciproka rummet, isoytan där elektronernas energi är lika med Fermi-energin. [10]
- **Git** är ett decentraliserat versionshanteringssystem. [5]
- **GUI** (Graphical User Interface) är ett grafiskt användargränssnitt. [11]
- **HDF5** är ett filformat som kan hantera stora mängder data [6]. Alla HDF5-objekt har en rotgrupp som äger alla andra objekt i datastrukturen. Denna grupp innehåller i sin tur all övrig data i form av andra grupper, länkar till andra grupper eller dataset. Dataset innehåller rådata av något slag. Rådata kan i sammanhanget vara bilder, utdata från beräkningar, programdata, etc. [7, s. 4-5]
De övriga objektstyperna går inte igenom i detalj i detta dokument, men finns väl beskrivna i *High Level Introduction to HDF5* [7].
- **Reciproka rummet**, även känd som k -rummet, är en fouriertransformering av det normala rummet. I reciproka rummet motsvarar varje punkt ett visst k -värde för en partikel. [12]

- **VASP** är ett program för modellering på atomnivå, för t.ex. elektronstrukturberäkningar och kvantmekanisk molekylodynamik. [1]
- **Rastergrafik** är en typ av data som innehåller en matris av färgvärden, med bestämd och informationsbegränsad upplösning. [13]

2 Översikt av systemet

Systemet är ett visualiseringsverktyg som kan användas för att visualisera utdata från elektronstrukturberäkningar i beräkningsprogrammet VASP. Programmet är modulärt programmerat och styrs via ett API.

2.1 Resurser

Följande resurser har använts för att utveckla systemet:

Mjukvara:

- Inviwo
- h5py
- Qt Creator
- CMake
- Git

Programmeringsspråk:

- Python 3
- C++ 14

2.2 Ingående delsystem

Delsystem 1 - System för parsning

Detta delsystem har i uppgift att läsa in data från VASP och omvandla till HDF5-format.

Delsystem 2 - System för visualisering

Detta delsystem har i uppgift att visualisera data som parsats av delsystem 1.

2.3 Utvecklingsfilosofi

Projektet har drivits med hjälp av versionshanteringssystemet Git och koden är licensierad med BSD 2, men även utvecklad under Inviwos utvecklaravtal för att, om önskvärt, kunna officiellt integreras i programvaran.

3 Delsystem 1 - System för parsning

Delsystemet 1 för parsning består av ett pythonbibliotek av moduler för att kunna hantera de olika utdatafilerna från VASP. Varje modul används för att läsa in specifika data från varje enskild utdatafil. Systemet har också en modul som anropas av samtliga andra moduler för att skriva HDF5-filer. Delsystemet parsar alltså först utdatafiler från VASP och strukturerar sedan om denna data till HDF5-format. Detta dataformat är delsystem 2 implementerat för att ha som indata, bland annat därför att data i detta format har en enklare struktur.

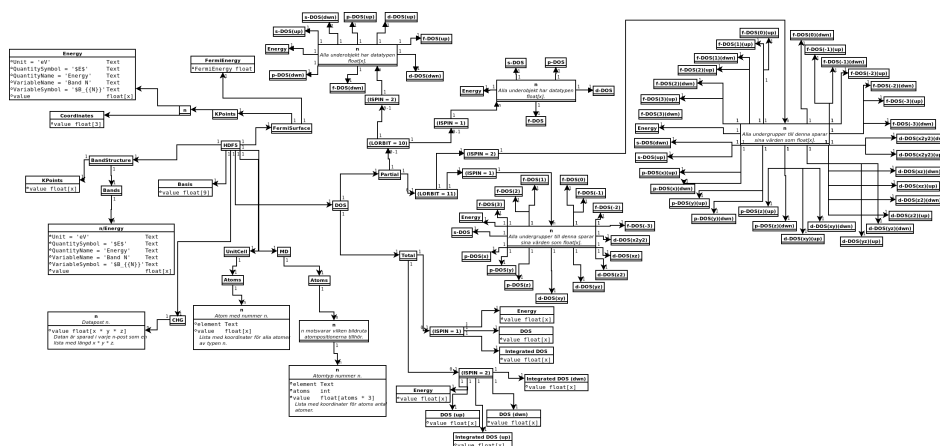
3.1 VASP

Från beräkningsprogrammet VASP fås en rad olika utdatafiler och dessa listas nedan.

- POSCAR innehåller data för enhetscellen samt atompositionsdata.
- CHG innehåller laddningstäthetsdata.
- DOSCAR innehåller tillståndstäthetsdata. Se kapitel 3.4.3 för mer information.
- EIGENVAL innehåller data för alla energier i k-rummet.
- POTCAR innehåller data om atomtyper.
- OUTCAR innehåller all utdata. I detta fall är information om Fermi-energi, den irreducibla 1:a Brillouin-zonens basvektorer och den 1:a Brillouin-zonens symmetriegenskaper vad som söks.
- XDATCAR innehåller data om enhetscell, atompositionsdata för varje beräkningssteg och även atomtyp.
- CONTCAR på samma format som POSCAR men CONTCAR fylls med information om atompositioner uppdaterats.

3.2 HDF5-struktur för VASP-data

För att enkelt kunna läsa data i Inviwo följer HDF5-filerna som skapas av parsersystemet filstruktur enligt figur 1 nedan.



I Figur 1 finns det HDF5-format som VASP-datan ska läsas in till beskrivet. En ruta i diagrammet motsvarar en del i sökvägen i HDF5-objektet, om inte titeln är inom parenteser, då det innebär att rutans barnnoder blir åtkomliga om jämförelsen inom parenteserna är sann. Om en ruta har namngivna fält så innebär fältet *value* den data som är sparad i ett dataset, medan övriga datafält är attribut. En ruta som heter *n*, eller har det i namnet, innebär att det finns flera dataset och att de är numrerade med heltal.

På grund av platsbrist saknar Figur 1 attribut för DOS-datan, men alla fält som innehåller DOS-data i diagrammet, till exempel p-DOS, d-DOS(xy), Energy, etc., har attribut som beskriver formatet på datan i fältet. En lista över attributen följer nedan:

- **VariableName** är fältets namn.
- **VariableSymbol** är en symbol som representerar variabeln.
- **QuantityName** är ett för en människa läsligt namn på fältet.
- **QuantitySymbol** är symbol som representerar storheten.
- **Unit** är storhetens fysikaliska enhet.

Den högra texten i varje ruta avser alltid datatypen för en datapost. I de fall datatypen följs av hakparanteser, till exempel *float[x]*, betyder det att datan är sparad i en lista där värdet inom hakparanteserna utgör listans längd.

Som exempel så koms tillståndstätheten för den totala tillståndstätheten åt via `/DOS/Total/DOS` oberoende av LORBIT. Ett till exempel är första energin för första bandet i bandstrukturen som nås via `/BandStructure/Bands/0/Energy`.

3.3 Modul för skrivning till HDF5

Denna modul består av en pythonfil med namnet h5writer innehållandes funktionerna `_write_coordinates`, `_write_basis`, `_write_md`, `_write_steps`, `_write_bandstruct`, `_write_dos`, `_write_volume` samt `_write_incar`. Dessa funktioner skapar grupper och dataset enligt figur 1 ovan. Nedan följer en kort beskrivning av varje funktion.

`_write_coordinates`

Denna funktion skriver koordinater för atompositioner där varje atomslag tilldelas ett eget dataset. Attribut sätts för respektive grundämnesbeteckning per dataset.

Parametrar:

- `h5file`: Sökväg till HDF5-fil.
- `atom_count`: Lista med antalet atomer av de olika atomslagen.
- `coordinates_list`: Lista med koordinater för samtliga atomer.
- `Elements`: = None eller lista med atomslag

Returnerar:

- None.

`_write_basis`

Denna funktion skriver gittervektorerna i ett dataset med namn `basis`.

Parametrar:

- `h5file`: Sökväg till HDF5-fil.
- `basis`: Lista med basvektorerna.

Returnerar:

- None.

`_write_bandstruct`

Denna funktion skriver ut data för bandstruktur i en grupp med namn `Bandstructure`. Inom denna grupp tilldelas specifika K-punkter, energier samt bandstrukturer egna dataset. Diverse attribut sätts även för bl.a. specifika energier.

Parametrar:

- `h5file`: Sökväg till HDF5-fil.
- `band_data`: Lista med bandstrukturdata.
- `kval_list`: Lista med K-punkter för specifika bandstrukturdata.

Returnerar:

- None.

`_write_dos`

Denna funktion skriver ut DOS-data i en grupp med namn `DOS` där total och partiell DOS tilldelas grupper med namn `Total` respektive `Partial`. Inom gruppen `Total` tilldelas energin samt specifika DOS egna dataset och inom gruppen `Partial` tilldelas varje partiell DOS egna grupper där energin samt specifika DOS tilldelas egna dataset.

Parametrar:

- `h5file`: Sökväg till HDF5-fil.
- `total`: En lista med strängar av de olika uträkningarna som har utförts av VASP för total DOS.
- `partial`: En lista med strängar av de olika uträkningarna som har utförts av VASP för partiell DOS.
- `total_data`: En lista med alla beräkningar för total DOS för varje specifik atom.
- `partial_list`: En lista med alla beräkningar för partiell DOS för varje specifik atom.
- `fermi_energy`: Fermi-energin för den aktuella uträkningen.

Returnerar:

- None.

`_write_volume`

Denna funktion skriver ut elektrontäthetsdata och elektronlokaliseringsfunktionsdata (ELF) till grupper med namn Charge respektive Elf. Inom dessa grupper tilldelas varje iteration ett dataset.

Parametrar:

- `h5file`: Sökväg till HDF5-fil.
- `i`: Skalar som anger numret på iterationen.
- `array`: Array med parsad data för respektive iteration.
- `data_dim`: Lista som anger dimensionen av data för respektive iteration.
- `hdfgroup`: En textsträng med namnet på vad man vill kalla gruppen i HDF5-filen.

Returnerar:

- None.

`_write_incar`

Denna funktion skriver ut parsad data från INCAR i ett dataset med namn Incar där varje datatyp tilldelas egna dataset.

Parametrar:

- `h5file`: Sökväg till HDF5-fil.
- `incar_data`: Datalexikon med all data från INCAR-filen.

Returnerar:

- None.

3.4 Moduler för parsning

Samtliga moduler för parsning består av diverse pythonfunktioner som utför själva parsningen av data. Dessa innefattar parsers som endast parsar specifika data, en som parsar INCAR-data samt en funktion som anropar samtliga parsers utom INCAR-parsern.

3.4.1 Incarparser

Incarparsern består av pythonfil med namnet `incar` som innehåller funktionerna, `incar` och `parse_incar`. Dessa funktioner läser in och sparar information från INCAR-filen samt anropar en separat pythonmodul som skriver en HDF5-fil. INCAR är en inputfil för VASP som anger hur VASP ska exekvera elektronberäkningar. VASP-användaren har initialt satt värden rad för rad för en mängd variabler.

Funktionen `incar` kontrollerar om HDF5-filen redan innehåller INCAR-data och anropar funktionen `parse_incar` om så inte är fallet. Existerar INCAR-filen i användarens VASP-katalog parsas data av funktionen `parse_incar` som då sparar ett dataset för varje datatyp och namnger dataseten därefter. Funktionen `incar` anropar sedan pythonmodulen som skriver HDF5-filen där varje enskilt dataset tilldelas en egen grupp (se delavsnitt 3.2 ovan för beskrivning av strukturen i HDF5-filen).

Funktionsanrop: `envision.parser.vasp.incar(h5file, vasp_dir)`

Parametrar:

- `h5file`: Sökväg till HDF5-fil.
- `vasp_dir`: Sökväg till VASP-katalog.

Returnerar:

- Lista med namn på data (dataseten) som parsats.
- Bool: True om parsning skett felfritt, False annars.

3.4.2 Volymparser

Volymparsern består av en mängd funktioner i en pythonfil som används för parsning av CHG och ELFCAR. Den kan läsa in och spara data på HDF5-format från båda dessa filer genom att anropa en pythonmodul. Detta är för att CHG och ELFCAR har samma struktur och består av ett antal iterationer av volymdata från volymberäkningar. Således innehåller den sista iterationen data som är mest korrekt. Därför skapar volymparsern också en länk till den sista iterationen i HDF5-filen för att data av högst kvalitet lätt ska kunna plockas ut.

Funktionsanrop vid parsning av CHG-data: `envision.parser.vasp.charge(h5file, vasp_dir)`

Funktionsanrop vid parsning av ELFCAR-data: `envision.parser.vasp.elf(h5file, vasp_dir)`

Parametrar:

- `h5file`: Sökväg till HDF5-fil.
- `vasp_dir`: Sökväg till VASP-katalog.

Returnerar:

- Bool: True om parsning skett felfritt, False annars.

3.4.3 Tillståndstäthetsparser

Tillståndstäthetsparsern består av en mängd funktioner i en pythonfil som används för parsning av DOSCAR. DOSCAR-filen består först av den totala tillståndstätheten och sedan partiell tillståndstäthet för varje atom i kristallen. Beroende på vad som står i INCAR kan dock denna data se väldigt olika ut. Flaggorna ISPIN, RWIGS och LORBIT i INCAR-filen avgör vad som skrivs i DOSCAR-filen. ISPIN-flaggan informerar om spinn har tagits hänsyn till vid beräkningar, RWIGS-flaggan specificerar Wigner-Seitz-radien för varje atomtyp och LORBIT-flaggan (kombinerat med RWIGS) avgör om PROCAR- eller PROOUT-filer (som DOSCAR-filen refererar till) skrivs. Parsern läser därför från data givet av incarparsern i HDF5-filen för att se hur DOSCAR ska parsas. Parsern delar upp data i två grupper i HDF5-filen, total och partiell. I gruppen partiell finns det en grupp för varje atom. Ett dataset för varje undersökt fenomen skrivs sedan ut för varje atom under partiell, och för total tillståndstäthet under total.

Funktionsanrop: `envision.parser.vasp.dos(h5file, vasp_dir)`

Parametrar:

- `h5file`: Sökväg till HDF5-fil.
- `vasp_dir`: Sökväg till VASP-katalog.

Returnerar:

- Bool: True om parsning skett felfritt, False annars.

3.4.4 Fermi-yteparser

Fermi-yteparsern består av en mängd funktioner i en pythonfil som används för parsning av OUTCAR och EIGENVAL. OUTCAR-filen parsas för att hitta Fermi-energin, EIGENVAL-filen parsas för varje k-punkts energivärden för varje band.

Funktionsanrop: `envision.parser.vasp.fermi(h5file, vasp_dir)`

Parametrar:

- `h5file`: Sökväg till HDF5-fil.
- `vasp_dir`: Sökväg till VASP-katalog.

Returnerar:

- Bool: True om parsning skett felfritt, False annars.

3.4.5 Enhetscellsparser

Enhetscellparsern läser in gittervektorer, som multipliceras med skalfaktorn och skrivs till /basis i HDF5-filen. Atompositioner läses från POSCAR och om dessa är angivna med kartesiska koordinater räknas de om till koordinater med gittervektorerna som bas. Koordinaterna skrivs till HDF5-filen uppdelade efter atomslag och attribut sätts med respektive grundämnesbeteckning. Om dessa inte ges med parametern `elements` letar parsern i första hand i POTCAR och i andra hand i POSCAR.

Funktionsanrop: `envision.parser.vasp.unitcell(h5file, vasp_dir, elements = None)`

Parametrar:

- `h5file`: Sökväg till HDF5-fil.
- `vasp_dir`: Sökväg till VASP-katalog.
- `elements = None`: None eller lista med atomslag.

Returnerar:

- Bool: True om parsning skett felfritt, False annars.

3.4.6 `parse_all`

`parse_all` är en funktion för parsning av allt som finns i katalogen som ges som inparameter. Funktionen kallar på alla systemets parsers och skriver ut meddelande om vad som parsas och om parsningen gjordes eller ej. Den returnerar en lista med alla namn på grupper i HDF5-filen eller None om HDF5-filen ej skapats.

Funktionsanrop: `envision.parse_all(h5_path, dir)`

Parametrar:

- `h5_path`: Sökväg till HDF5-fil.
- `vasp_dir`: Sökväg till katalog med utdata-filer från beräkningsprogram.

Returnerar:

- Bool: True om parsning skett felfritt, False annars.

4 Delsystem 2 - System för visualisering

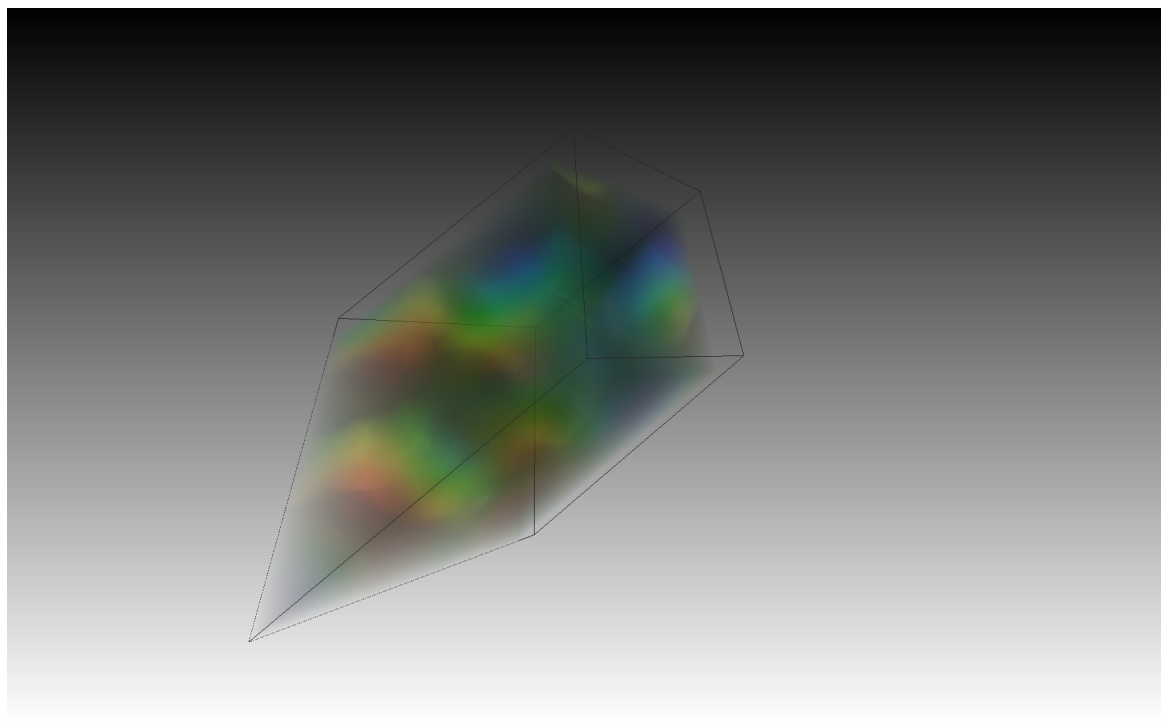
Visualiseringssystemet består av ett antal pythonmoduler för nätverksbygge i Inviwo samt c++-moduler implementerade i Inviwo av projektgruppen för att visualisera data.

4.1 Nätverk

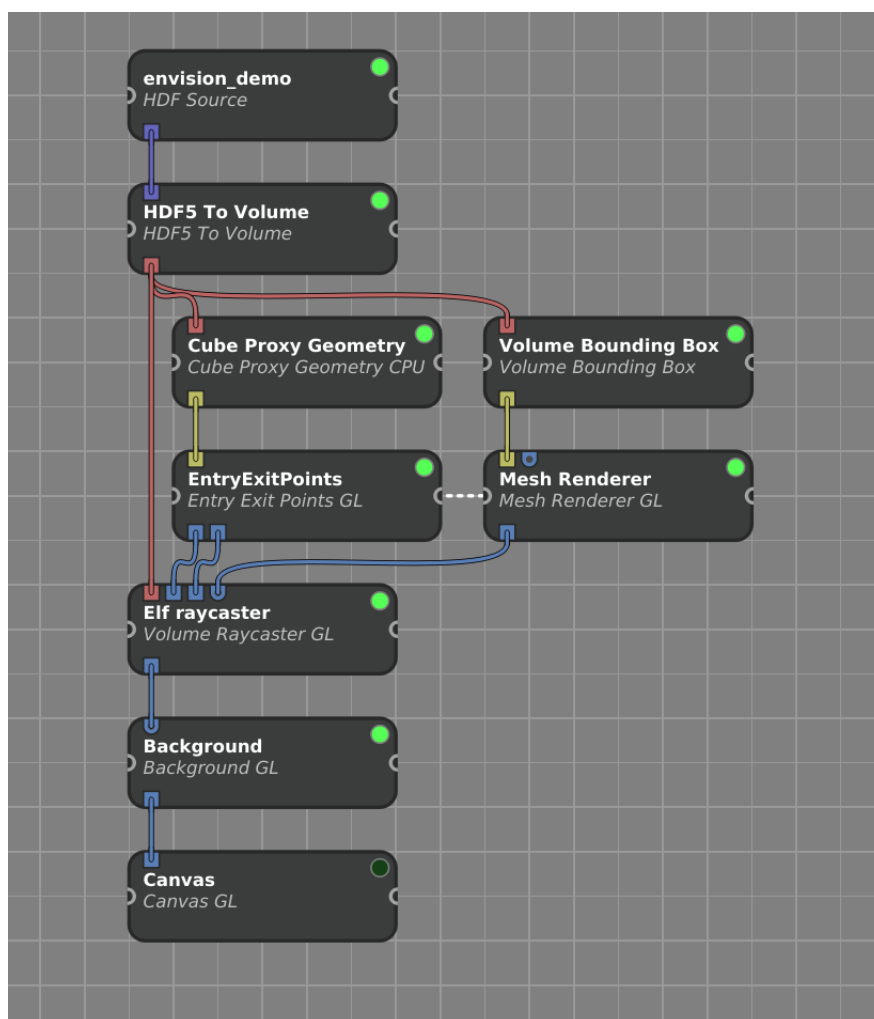
Processorer är de primära objekt i nätverken som användaren interagerar med. De består bland annat av inportar och utportar som används vid utbyte av data av specifika typer. Nätverken mynnar ut i en så kallad Canvas, en processor som på sin inport tar rastergrafikdata som den visualiserar i ett separat fönster. Samtliga pythonmoduler som bygger nätverk består i huvudsak av kod som anropar Inviwos egna interna funktioner för att bland annat importera, koppla, länka samt sätta värden på specifika parametrar i processorer.

4.1.1 Nätverk för visualisering av volymdata

Data som visualiseras med hjälp av volymnätverket är elektrontäthetsdata och elektronlokaliseringssfunktionsdata (ELF-data). Nätverket kan visualisera data med hjälp av tekniken volume ray casting eller iso ray casting. Se figur 2 nedan som visar en skärmbild från Inviwo när visualisering av ELF-data för diamant med iso ray casting görs. Det tillhörande nätverket syns i figur 3.

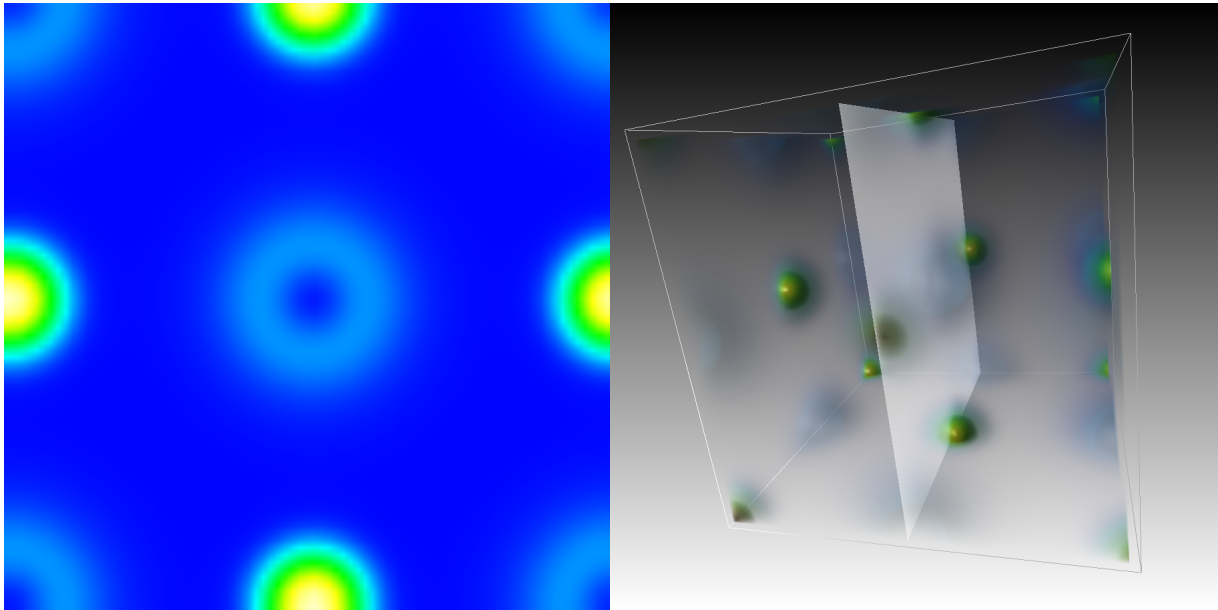


Figur 2: Visualisering av ELF-data för diamant i Inviwo.

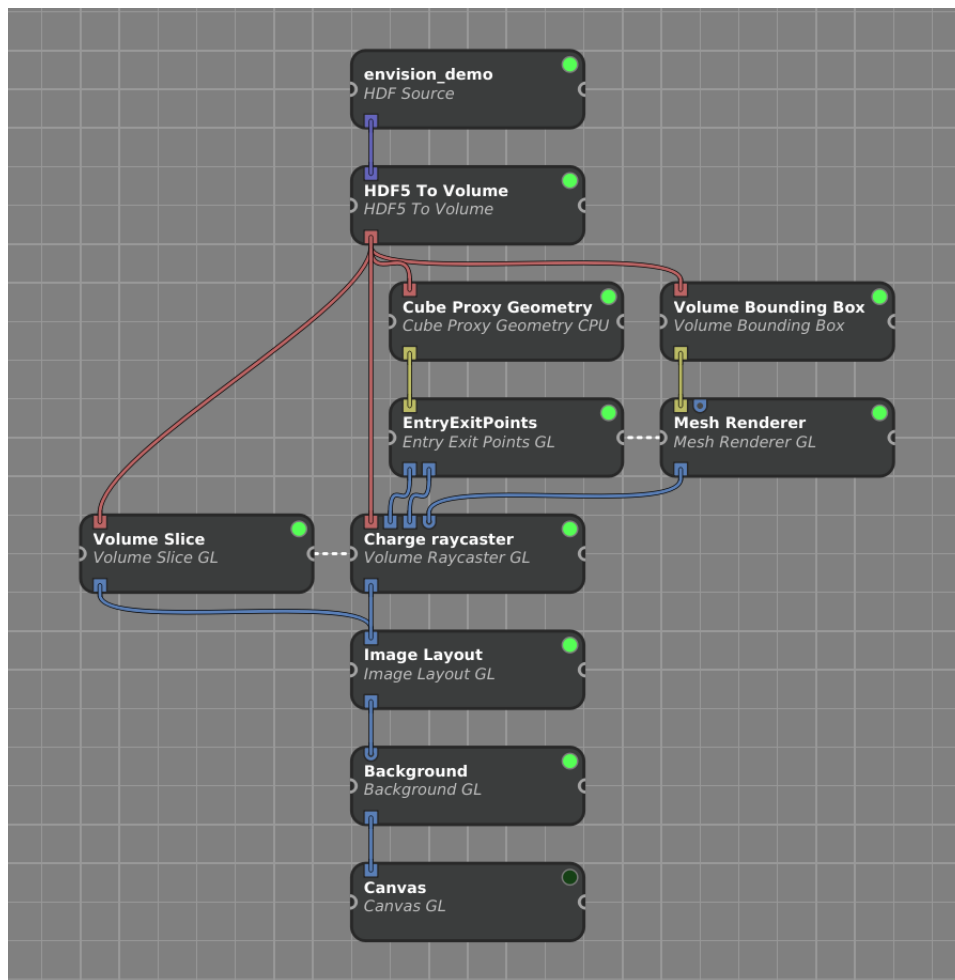


Figur 3: Nätverket för visualisering av ELF-data för diamant i Inviwo

När tekniken volume ray casting används har användaren möjlighet att aktivera en slice-funktion som skapar ett plan som skär igenom volymen. Visualiseringen består då av två stycken bilder placerade bredvid varandra i samma Canvas. Högra bilden visar hela volymen tillsammans med ett plan, som användaren på eget behag kan flytta, och vänstra bilden visar utseendet av det specifika tvärsnitt av volymen som planet i varje enskild position skapar. Se figur 4 nedan som visar en skärmbild från Inviwo när visualisering av laddningstäthet för natriumklorid, NaCl, med volume ray casting och slice-funktion görs, och figur 5 för tillhörande nätverk. Försöker användaren visualisera data med iso ray casting och slice-funktion returneras meddelandet "Slice is not possible with ISO Raycasting, therefore no slice-function is showing." och en visualisering med iso ray casting utan slice-funktion görs.



Figur 4: Visualisering av laddningstäthet för NaCl med slice-funktion i Inviwo.



Figur 5: Nätverket för visualisering av laddningstäthet för NaCl med slice-funktion i Inviwo.

Systemet som bygger volymnätverket består av en pythonfil med namnet volume innehållandes

tre funktioner: charge, elf och volume_network.

Funktionerna charge och elf är de funktioner som användaren anropar och de tar vardera fem argument:

- h5file: Sökvägen till HDF5-filen där samtlig data finns.
- iso: Önskas iso-yta sätts denna till det specifika värdet iso-ytan ska ha annars till None. Defaultvärde = None.
- slice: Bool-variabel som sätts till True om slicefunktion önskas, annars till False.
- xpos: Den översta processorns x-koordinat. Default = 0.
- ypos: Den översta processorns y-koordinat. Default = 0.

Funktionsanrop charge:

```
envision.inviwo.charge(h5file, iso = None, slice = False, xpos = 0, ypos = 0)
```

Funktionsanrop ELF:

```
envision.inviwo.elf(h5file, iso = None, slice = False, xpos = 0, ypos = 0)
```

Dessa två funktioner anropar funktionerna volume_network som bygger själva nätverket. Denna tar argumenteten:

- h5file: Sökvägen till HDF5-filen där samtlig data finns.
- volume: En textsträng som anger om elektrontäthetsdata eller ELF-data ska visualiseras.
- iso: Önskas iso-yta sätts denna till det specifika värde iso-ytan ska ha annars till None.
- slice: Bool-variabel som sätts till True om slicefunktionen önskas, annars False.
- xstart_pos: Den översta processorns x-koordinat.
- ystart_pos: Den översta processorns y-koordinat.

Funktionen charge anropar användaren för att visualisera elektrontäthet och funktionen elf för att visualisera ELF-data. Båda dessa anger argumentet "h5file" i funktionen volume_network med det värde användaren gett som argument "h5file" till respektive funktion som. Argumentet "volume" anger funktionen charge som "Charge raycaster" och funktionen elf som "Elf raycaster". Resterande argument sätts, på samma sätt som för argumentet "h5file", till det värde användaren gett motsvarande argument till respektive funktion som.

Funktionen volume_network laddar en HDF Source processor som ges samma namn som HDF5-filen. Dess utdata skickas till en HDF5 To Volume processor. HDF Source processorn anger vilken HDF5-fil och HDF5 To Volume processorn anger, baserat på argumentet "volume", vilken specifik grupp i HDF5-filen som data ska laddas från. Utdata från HDF5 To Volume processorn skickas till processorerna Cube Proxy Geometry, Volume Bounding Box samt till processorn för ray casting. Baserat på argumentet "iso" laddas processorn ISO Raycaster eller Volume Raycaster och denna ges namnet "Charge raycaster" eller "Elf raycaster" baserat på argumentet "volume". Processorn Cube Proxy Geometry skapar en proxy geometri i form av en mesh formad som en parallelepiped och skickar denna data till processorn Entry Exit Points. Denna

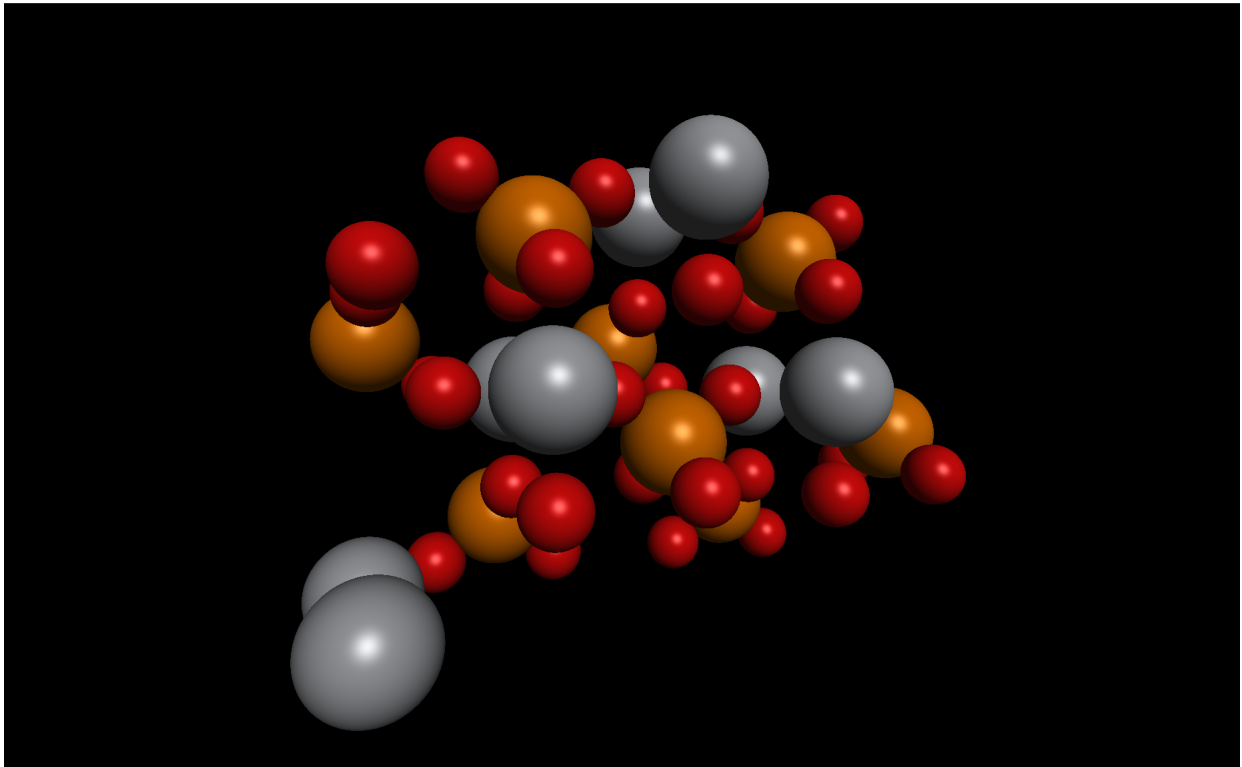
processor beräknar start-och slutpunkt för respektive ”stråle där data samplas från” i processorn för ray casting och utgör således en del av denna processors indata. Beräkningen av start- och slutpunkt görs utgående från aktuell vinkel som den kamera som ”ser in i” volymen data representerar har. Processorn Volume Bounding Box definierar geometrin för en mesh som omsluter volymen (utgör dess kanter) och skickar denna data till processorn Mesh Renderer som skapar meshen. Mesh Renderer skickar sedan denna data till processorn för ray casting.

Baserat på utdata från processorerna HDF5 To Volume, Entry Exit Points och Mesh Renderer skapar sedan processorn för ray casting en 2-dimensionell representation av volymen som data representerar. Denna data skickas, via processorn Background som lägger till en bakgrund, till Canvas-processorn. Se figur 3 som visar en skärmbild från Inviwo över nätverket när ELF-data för diamant visualiseras med volume ray casting.

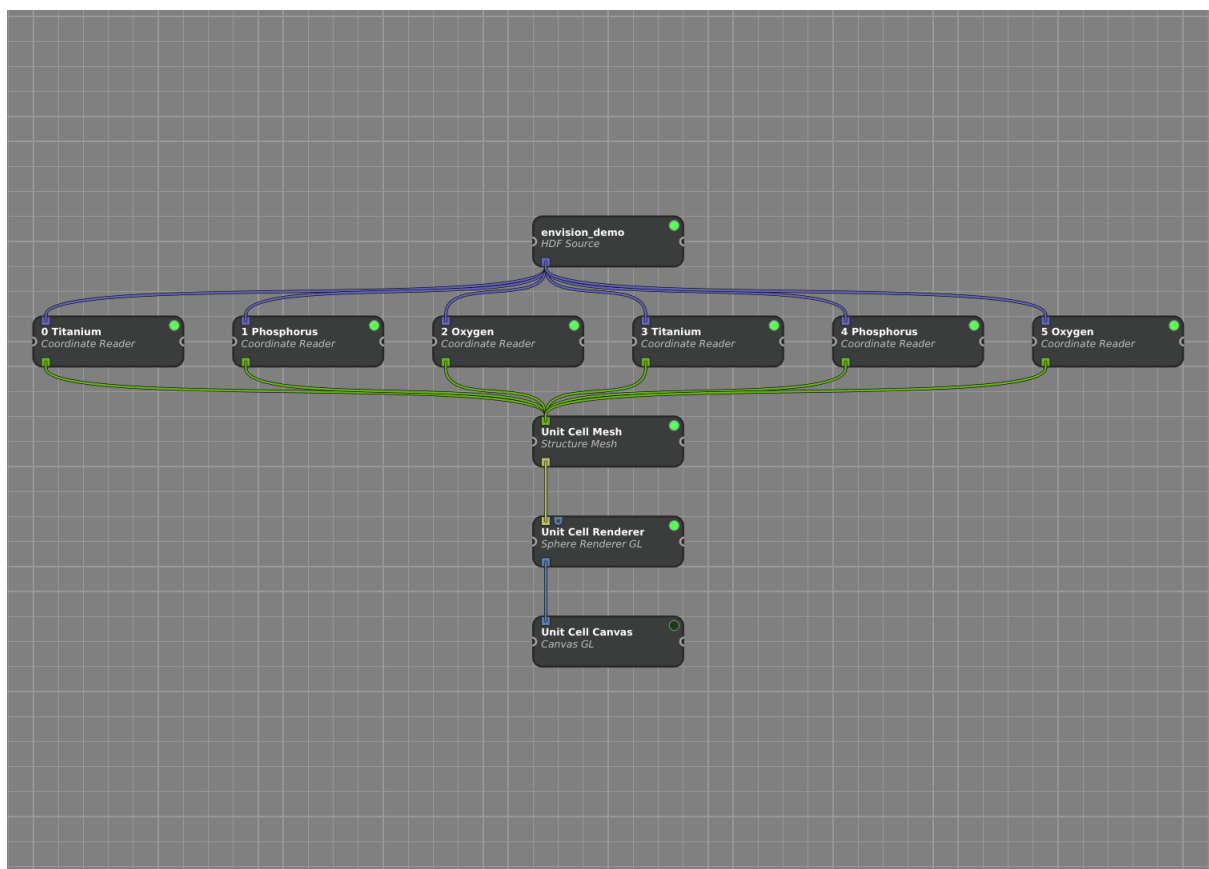
Om slicefunktionen används laddar även funktionen volume_network en Volume Slice processor och en Image Layout processor. Volume Slice processorn skapar det plan som skär volymen se figur 4 till höger. Indata till denna processor utgörs av utdata från HDF5 To Volume processor och dess utdata går till processorn Image Layout. Processorn Image Layout skapar en delad vy över de båda bilderna se figur 4 ovan och dess indata utgörs, förutom av utdata från Volume Slice processorn, av utdata från processorn för ray casting. Denna data skickas sedan, via processorn Background som skapar en bakgrund, till Canvas-processorn. Se figur 3 som visar en skärmbild från Inviwo över nätverket när laddningstäthet för natriumklorid, NaCl, visualiseras med hjälp av volume ray casting och slice-funktion.

4.1.2 Nätverk för visualisering av enhetscell

Funktionen för visualisering av enhetscell placerar ut en koordinatläsarprocessor (Coordinate-Reader) per atomslag och kopplar dessa till en processor av typen HDF source, som läser den av användaren angivna HDF5-filen. Utdata från koordinatläsarna skickas till en StructureMesh-processor som placerar ut klot som representerar atomer. Färg och radie sätts enligt en fördefinierad lista, men kan ändras via Inviwos grafiska användargränssnitt. Utdata från StructureMesh skickas till processorn Sphere Renderer, som sköter själva renderingen. Utporten på denna processor kan kopplas till den blå inporten på processorn Mesh Renderer i volymrenderingsnätverket, se delavsnittet 4.1.1 om det ta nätverk ovan, så att enhetscellen visas tillsammans med elektronstrukturen. Se figur nedan som vardera visar en skärmbild från Inviwo när visualisering av enhetscell för respektive visualisering av enhetscell sammankopplat med visualisering av ELF data med volume ray casting för görs.



Figur 6: Visualisering av TiPO4 i Inviwo



Figur 7: Nätverket för visualisering av TiPO4 i Inviwo.

4.1.3 Nätverk för visualisering av DOS

Nätverket för visualisering av tillståndstäthetsdata laddar en HDF source processor som anger HDF5-filen som data laddas från om HDF source processor inte finns, annars kopplas en HDF5 Path Selection processor direkt till den redan befintliga HDF source processorn. Denna processor anger att data ska laddas från DOS-gruppen i HDF5-filen. Två till HDF5 Path Selection processorer laddas sedan som anger grupperna Total och Partial i HDF5-filen.

För Total-delen laddas sedan kontinuerligt HDF5ToFunction processorer som gör funktioner av all data i Total-gruppen. För Partial-gruppen laddas en HDF5 Path Selection All Children processor som tar ut alla dataset för alla atomer. Därefter laddas HDF5ToFunction processorer för alla dataset i grupperna under Partial-gruppen. All data matas sedan i en plotter-processor som gör en graf i matplotlib, omvandlar grafen till rå bilddata och matar in detta i en Canvas-processor som visar själva grafen. Se figur nedan som visar en skärmbild från Inviwo över nätverket när tillståndstäthet visualiseras för och i figur nedan visas en graf över

4.2 Datastrukturer

Två datastrukturer, Point och Function, har introducerats. Dessa används i vissa av de implementerade processorerna.

4.2.1 Point

Denna datatyp representerar en reell 1D-punkt och inkapslar punktens värde (ett fluttal) samt variabel metadata.

4.2.2 Function

Denna datatyp representerar en reellvärd funktion av en reell variabel och inkapslar sampelvärden och variabel-metadata för x- och y-axlarna.

4.3 Processorer

Ett antal processorer har implementerats, dessa kategoriseras och beskrivs nedan.

4.3.1 Kristallstruktur

Nedanstående processorer är relaterade till visualiseringen av kristallstrukturer.

CoordinateReader

Från en HDF5-fil läser denna processor koordinater för atompositioner. En sökväg till ett dataset sätts via en property kallad StringProperty. Denna StringProperty ska ha storleken $n*3$ och bestå av 32-bitars fluttal. Utdata från CoordinateReader är n stycken `vec3`.

Inport:

- `Hdf5::Inport inport_`

Utport:

- DataOutput< std::vector<vec3> > output_

Properties:

- StringProperty path_

StructureMesh

Atompositionsdata kopplas ihop med rätt atomfärg och radie med StructureMesh-processorn. Dessutom hanterar processorn molekylodynamik och "picking", funktionen som gör det möjligt för användaren att välja atomer genom att klicka på dem. StructureMesh har en multiimport, dit en eller flera CoordinateReader-processorer kan kopplas in. Indata för StructureMesh är atompositionsdata i form av vec3 för varje atomslag. Till denna indata läggs properties för färg, radie och antal till för varje atomslag/processor som kopplas in. Den ger en mesh, som har buffrar för position, färg och radie.

Inport:

- DataInput< std::vector<vec3>, 0> structure_

Utport:

- MeshOutput mesh_

Properties:

- FloatProperty scalingFactor_
- FloatMat3Property basis_
- BoolProperty fullMesh_
- BoolProperty animation_
- IntProperty timestep_
- std::vector< std::unique_ptr<FloatVec4Property> > colors_: vektor som innehåller färg-property för varje atomslag
- std::vector< std::unique_ptr<FloatProperty> > radii_: vektor som innehåller radieproperty för varje atomslag
- std::vector< std::unique_ptr<IntProperty> > num_: vektor som innehåller antalet atomer per tidssteg för varje atomslag
- IntVectorProperty inds_: vektor med index på valda atomer

4.3.2 HDF5

Nedanstående processorer är ämnade att fungera väl med de HDF5-relaterade processorer som är inkluderade i Inviwo.

HDF5PathSelection*

Detta är en grupp av processorer som har funktionalitet liknande den inbyggda processorn `HDF5PathSelection`. En eller flera av dessa processorer placeras med fördel mellan en `HDFSource` och en eller flera `HDF5To*`.

Gemensamt för dessa processorer är att de på inporten tar en `Hdf5`-grupp och på utporten skriver noll eller flera av dessa omedelbara undergrupper.

Nedan beskrivs de olika processorerna i denna grupp.

HDFpathSelectionInt

Denna processor väljer en `HDF5`-grupp med heltalsnamn, baserat på värdet på processorns `intProperty_`, eventuellt utäkat med ledande nollor till bredden specificerat på processorns `zeroPadWidthProperty_`.

`HDF5PathSelectionInt` kan med fördel användas tillsammans med en `OrdinalPropertyAnimator` för att plocka ut relevant data ur en `HDF5`-fil.

Anledningen till att utdata ges som en vektor av `HDF5`-grupper, trots att processorn alltid skriver exakt en grupp på utporten, är att processorn ska följa samma mönstersom, och fungera väl med, resterande processorer.

Inport:

- `DataInport<hdf5::Handle> hdf5HandleInport_`

Utport:

- `DataOutput< std::vector<hdf5::Handle> > hdf5HandleVectorOutput_`

Properties:

- `IntProperty intProperty_`
- `IntSizeTProperty zeroPadWidthProperty_`

HDF5PathSelectionIntVector

Denna processor väljer noll eller flera `HDF5`-grupper med heltalsnamn, baserat på värdet på processorns `intVectorProperty_`, eventuellt utökat med ledande nollor till berdden specificerat av processorns `zeroPadWidthProperty_`.

Inport:

- `DataInport<hdf5::Handle> hdf5HandleInport_`

Utport:

- `DataOutput< std::vector<hdf5::Handle> > hdf5HandleVectorOutput_`

Properties:

- IntVectorProperty intVectorProperty_
- IntSizeTPProperty zeroPadWidthProperty_

HDF5PathSelectionAllChildren

Denna processor väljer den givna HDF5-gruppens alla undergrupper.

Inport:

- DataInport<hdf5::Handle> hdf5HandleInport_

HDF5To*

Detta är en grupp av processorer som har funktionalitet liknande den inbyggda processorn HDF5ToVolume. Processorerna placeras med fördel efter en HDFSource-processor, med en eller flera mellanliggande HDF5PathSelection*.

Gemensamt för dessa är att de som indata tar noll eller flera HDF5-grupper (baserat på *pathSelectionProperty_), plockar ut dataset för varje grupp och omvandlar dessa till relevanta objekt (Point eller Function) som sedan skrivs till utporten. Objektens variabel-metadata tas, om de finns tillgängliga, från attributen associerade med dataseten. Vidare kan, om så väljs med *namePrependParentsProperty_, metadata utökas med namnen på de grupper vari dataseten ligger.

Vilka dataset som kan väljas med *pathSelectionProperty_ uppdateras dynamiskt beroende på vilka grupper som ligger på inporten. När ett lämpligt dataset valts kan *pathFreezeProperty_ användas för att stänga av denna dynamik, så att värdet sparas även om grupperna på inporten (antagligen tillfälligt) ändras. Detta underlättar manuellt experimenterande samt användandet av processorer som tillfälligt ger noll grupper som utadat, t.ex. HDF5PathSelectionIntVector.

HDF5ToPoint

Denna processor konverterar HDF5-data till noll eller flera Point-objekt.

Inport:

- DataInport<hdf5::Handle, 0, true> hdf5HandleFlatMultiInport_

Utport:

- DataOutport< std::vector<Point> > pointVectorOutport_

Properties:

- OptionPropertyString pathSelectionProperty_
- BoolProperty pathFreezeProperty_
- IntSizeTPProperty namePrependParentsProperty

HDF5ToFunction

Denna processor konverterar HDF5-data till noll eller flera Function-objekt.

Normalt plockas två dataset per grupp ut, ett för x-axeln och ett för y-axeln. Om endast data för y-axeln finns tillgänglig kan implicitXProperty_ sättas, varvid processorn automatgenererar data för x-axeln.

Inport:

- `DataInput<hdf5::Handle, 0, true> hdf5HandleFlatMultiInput_`

Utport:

- `DataOutput< std::vector<Function> > functionVectorOutput_`

Properties:

- `BoolProperty implicitXProperty_`
- `OptionPropertyString xPathSelectionProperty_`
- `OptionPropertyString yPathSelectionProperty_`
- `BoolProperty xPathFreezeProperty_`
- `BoolProperty yPathFreezeProperty_`
- `IntSizeTProperty xNamePrependParentsProperty_`
- `IntSizeTProperty yNamePrependParentsProperty_`

4.3.3 2D

Nedanstående processorer är ämnade att bearbeta och presentera 2D-data, närmare bestämt data av typen Point och Function.

FunctionOperationUnary

Denna processor implementerar en unär operator, antingen negation ($g_i(x) = -f_i(x)$) eller (multiplikativ) inversion ($g_i(x) = 1/f_i(x)$). Operatoren appliceras på funktioner på inporten, en i taget, och skriver respektive resultat på utporten.

Inport:

- `DataInput<Function, 0, true> functionFlatMultiInput_`

Utport:

- `DataOutput< std::vector<Function> > functionVectorOutput_`

Properties:

- `OptionPropertyString operationProperty_`

FunctionOperationNary

Denna processor implementerar en operator med variabel aritet (engelska n-ary), antingen addition/summa ($g(x) = \sum_i f_i(x)$) eller multiplikation/produkt ($g(x) = \prod_i f_i(x)$). Operatoren appliceras på samtliga funktioner på inporten och skriver resultatet på utporten.

Då funktionerna på inporten kan vara samplade vid olika x-värden behöver processorn ta beslut om var ut-funktionen ska samplas. Processorn utgår från att sampla i samtliga x-värden för

samtliga in-funktioner. `sampleFilterEnableProperty_` kan sättas för att filtrera dessa. Då `sampleFilterEnableProperty_` är satt ser processorn till att sampelavståndet är minst det värde som anges i `sampleFilterEpsilonProperty_`. När processorn skapas är `sampleFilterEnableProperty_` satt och `sampleFilterEpsilonProperty_` är 0 vilket innebär att x-värden som är identiska filtreras bort.

Om ett värde behöver beräknas vid ett x-värde där en in-funktion inte är samplat används linjär interpolation om x-värdet ligger innanför funktionens definitionsintervall. Om x-värdet ligger utanför detta intervall används `undefinedFallbackProperty_` för att avgöra vilket värde som används istället. Detta kan antingen vara noll eller funktionens värde vid intervallets relevanta ändpunkt.

Anledningen till att utdata ges som en vektor av funktioner, trots att processorn alltid skriver exakt en funktion på utporten, är att processorn ska följa samma mönster som, och fungera väl tillsammans med, resternade processorer.

Inport

- `DataInport<Function, 0, true> functionFlatMultiInport_`

Utport:

- `DataOutput< std::vector<Function> > functionVectorOutport_`

Properties:

- `OptionsPropertyString operationProperty_`
- `OptionsPropertyString undefinedFallbackProperty_`
- `BoolProperty sampleFilterEnableProperty_`
- `FloatProperty sampleFilterEpsilonProperty_`

4.4 Properties och widgets

4.4.1 IntVectorproperty

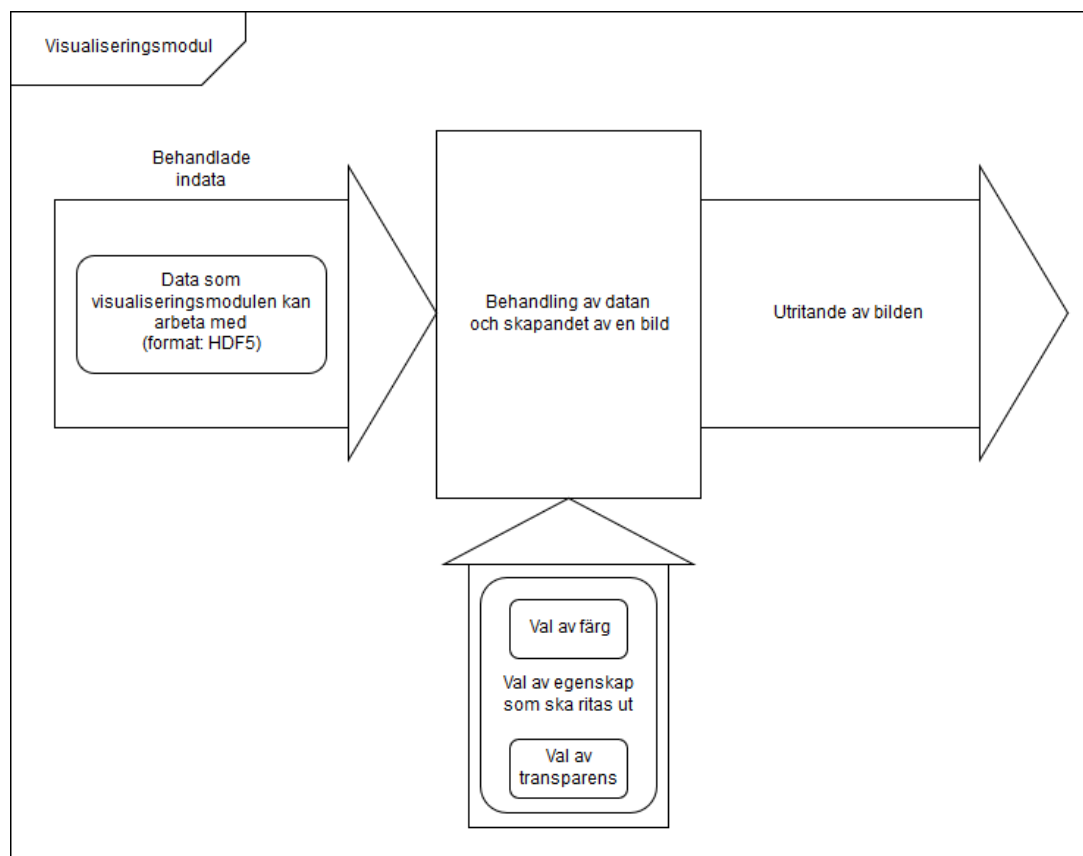
Denna property består av en vektor av int-värden.

4.4.2 IntVectorPropertyWidget

En widget för `IntVectorProperty`. "Textbox", satt till endast läsning (read only), som innehåller de värden som finns i tillhörande `IntVectorProperty`.

5 Visualisering

Figur 8 visar hur delsystemet Visualisering är uppbyggt. Behandlad samt användarindata skickas in. Användarindata består av val av färg, val av egenskap som ska ritas ut, val av transparens etc. Dessa behandlas sedan i visualiseringsmodulen som skapar en bild utifrån dem. Den utritade bilden skickas sedan ut från modulen. Detta är en allmän beskrivning av visualiseringen, för de olika egenskaperna som ska visualiseras kommer mittenrektangeln i Figur 8 att vara uppbyggd på olika sätt med olika processorer. Detta går mer in på i avsnitt 4.5-4.8.



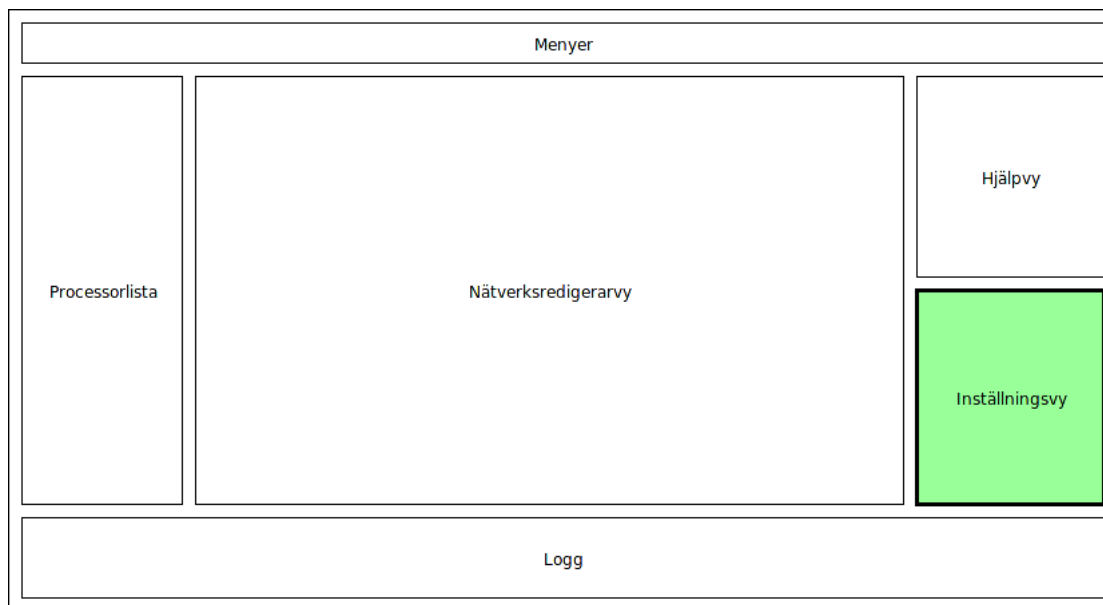
Figur 8: Grov design av delsystemet Visualisering

5.1 Utritning

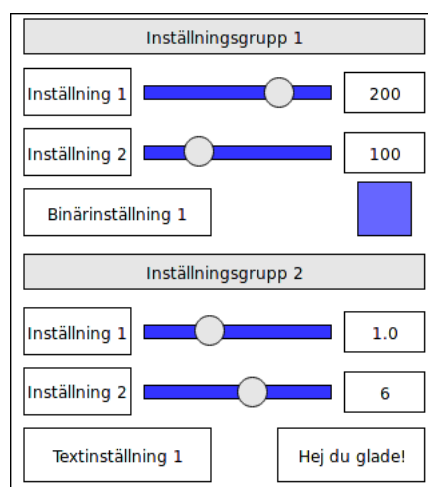
Delsystemet ska rita ut bilder utefter den behandlade datan. Utritningen skall ge en visualisering av den egenskap som modulen behandlar och kan till exempel vara en volymrendering eller en vanlig 2D-graf, beroende på vilket som är mest lättolkat.

Utritningen görs via Inviwos inbyggda funktionalitet för att rendera via OpenGL. Grafisk behandling görs också med Inviwo och OpenGL, där möjligheten finns att både använda färdig funktionalitet och att utveckla nya renderingsfunktioner.

5.2 Användarindata



Figur 9: Skiss som visar det tänkta användargränssnittet.



Figur 10: Skiss som visar den tänkta inställningsrutan i Inviwo.

Användaren kommer kunna ändra inställningar som kontrollerar utseendet på visualiseringarna. Denna indata matas in i Inviwos användargränssnitt, se Figur 9, och i inställningsrutan för en processor, se Figur 10.

Visualiseringsfunktionaliteten kommer att läggas till i form av processorer som via Inviwo tillhandahåller inställningsmöjligheter på det sätt som beskrivits ovan. Således blir inte användarinställningar ett eget separat delsystem, utan kommer att bakas in i visualiseringssystemen. Detta görs genom att processorerna skrivs att exponera de inställningar som användaren är menad att justera. Inviwo kommer då automatiskt lägga till inställningarna i rutan skisserad i Figur 10.

5.3 Interaktivitet

Användaren ska kunna modifiera visualiseringen genom att reglera ett intervall av värden för någon egenskap, där full transparens fås för alla värden inom intervallet, rotera 3D-bilder etc.

Ny användarindata, som skickas in efter att den första bilden har ritats upp, skickas tillbaka till visualiseringsmodulen för att utföra en ny rendering.

5.4 HDF5-indata

I Inviwo 0.9.9 finns en modul som tillåter importering av data från HDF5-fil [8]. Denna kommer användas för att ladda in data som sedan ska visualiseras. Se avsnitt 3.1 för beskrivning av uppbyggnaden av HDF5-filen.

5.5 Kristallstruktur

Kristallstruktur ska visualiseras som atompositioner i enhetscellen. Sfärer som representerar atomer ska ritas ut genom volymsrendering. Nedan följer en översiktlig beskrivning av hur detta ska byggas upp.

5.5.1 Översiktlig beskrivning

Koordinaterna (tre dimensioner) för atomerna läses in från HDF5-fil. Utifrån dessa positioner ritas sedan atomerna ut. Färginställningar för atomerna kan ändras av användaren.



Figur 11: Översiktligt flödesdiagram över dynamisk visualisering av kristallstruktur.

I Figur 11 beskrivs detta med ett flödesschema. Inviwo-nätverket näst sist i flödesschemat skickar ut en 3D-bild av kristallstrukturen där atomerna har placerats ut på sina positioner som hämtats från HDF5-filen.

5.5.2 Inviwo-nätverk för visualisering av kristallstruktur

En 3D-bild ska ritas ut vilket kommer kräva ett antal steg. Varje steg kommer att implementeras i Inviwo med en eller flera processorer.

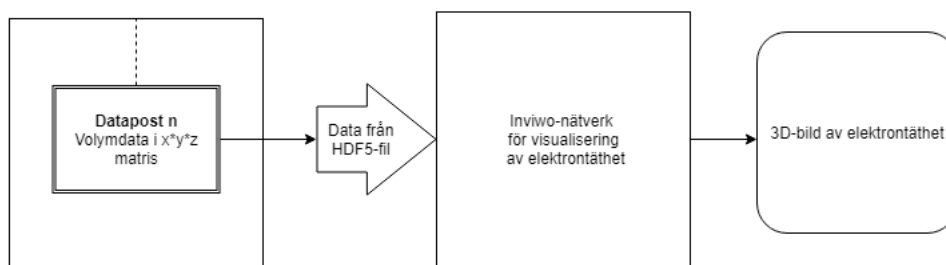
- HDF5-inläsning
 - Ladda in data från HDF5-fil.
- Koordinatläsning
 - Läs in koordinaterna för atomerna från HDF5-fil.
- Mesh-konstruering
 - Bygga upp ett mesh utifrån koordinaterna från koordinatläsar-processorn.
- Canvas
 - Visa 3D-bilden.

5.6 Elektrontäthet

Elektrontäthet visualiseras genom volymsrendering. Nedan följer en översiktlig beskrivning av hur detta ska byggas upp.

5.6.1 Översiktlig beskrivning

Datan som ritas ut är sannolikheten för att hitta elektroner på olika positioner. Sannolikheterna visualiseras med färger och skiftande transparens.



Figur 12: Översiktligt flödesdiagram över visualisering av elektrontäthet.

5.6.2 Inviwo-nätverk för visualisering av elektrontäthet

En 3D-bild ska ritas upp vilket kräver ett antal steg som implementeras med processorer i Inviwo.

- HDF5-inläsning
 - Ladda in data från HDF5-fil.
- Konvertering till volym

- Konvertera datan till en volym.
- Canvas
 - Visa 3D-bilden.

5.7 Tillståndstäthet

Tillståndstätheten, total och partiell, ska visualiseras med en 2D-graf. Nedan följer en översiktlig beskrivning av hur detta ska byggas upp.

5.7.1 Översiktlig beskrivning

Data hämtas från dataseten i gruppen tillståndstäthet i HDF5-filen. För partiell-tillståndstäthet hämtas energi, tillståndstäthet och så vidare för varje atom. Detta ska sedan plottas i en 2D-graf med tillståndstätheten på y-axeln och energin på x-axeln. Energin ska ha sitt nollställe vid Fermi-energin.



Figur 13: Översiktligt flödesdiagram över visualisering av tillståndstäthet.

I figur 13 beskrivs förloppet översiktligt. För partiell-tillståndstäthet hämtas alltså datan i första rektangeln i figuren för varje atom. För total-tillståndstäthet är detta dataseten för gruppen Total. Ett Inviwo nätverk för visualisering av tillståndstäthet måste skapas, detta nätverk illustreras i figur 13 av en stor kvadrat som skickar ut en 2D-graf på en canvas.

5.7.2 Inviwo-nätverk för visualisering av tillståndstäthet

En 2D-graf ska ritas upp vilket kräver ett antal steg. Dessa implementeras i Inviwo på en eller flera processorer.

- HDF5-inläsning
 - Ladda in data från HDF5-fil.
- Plottning
 - Plotta en graf av den laddade datan.

- Canvas
 - Visa den plottade grafen.

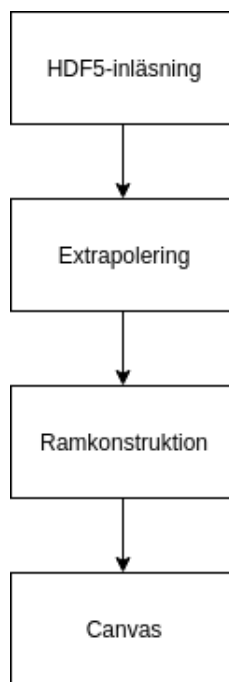
5.8 Fermi-ytor

Fermi-ytor och godtyckliga energi-isoytor ska kunna ritas upp i reciproka rummen, k -rummet. Detta görs genom att den 1:a Brillouin-zonen ritas upp och inuti denna ritas ytan.

5.8.1 Inviwo-nätverk för Brillouin-zonuppritning

Fermi-ytor ritas upp inuti 1:a Brillouin-zonen, som är den samling k -punkter i reciproka rummen vars närmsta gitterpunkt är origo. Den irreducibla 1:a Brillouin-zonen är den minsta samling vektorer som behövs för att extrapolera den fullständiga zonen, som kan byggas upp från kristallens symmetrigrupp. Brillouin-zonuppritningsnätverkets uppgift är att ta den irreducibla 1:a Brillouin-zonens basvektorer i kartesiska koordinater samt dess symmetriegenskaper, beräkna alla k -punkter som ingår i den 1:a Brillouin-zonen och rita upp en ram för denna. Basvektorerna parsas från OUTCAR-filen av en VASP-beräkning rörande Fermi-ytor, och återfinns på HDF5-format.

Inviwo-nätverket för att rita upp den 1:a Brillouin-zonen består av ett antal steg som beskrivs nedan. Se figur 14 för en skiss av hur dessa förhåller sig till varandra. Varje steg implementeras i Inviwo i form av en eller flera processorer.



Figur 14: Översiktlig beskrivning av de nödvändiga stegen för Brillouin-zonuppritning

- HDF5-inläsning
 - Hämtar data om basvektorerna för den irreducibla 1:a Brillouin-zonen
- Extrapolering
 - Använder basvektorerna och symmetrigruppen från HDF5-inläsningen för att extrapolera vilka punkter som ingår i den 1:a Brillouin-zonen

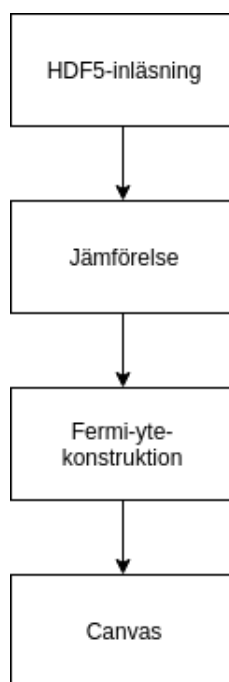
- Ramkonstruktion
 - Beräknar vilka k-punkter som utgör “ramen” för 1:a Brillouin-zonen
- Canvas
 - Ritar upp ramen

5.8.2 Inviwo-nätverk för Fermi-yte-uppritning

En sorts utdata från VASP-beräkningar är möjliga energier för alla k-punkter i reciproka rummet. Fermi-yte-uppritningsnätverkets uppgift är att ta dessa punkter med tillhörande energier och beräkna samt rita upp isoytan där energin är lika med Fermi-energin inuti ramarna för 1:a Brillouin-zonen. Energierna för alla k-punkter parsas från EIGENVAL-filen. Fermi-energin parsas från OUTCAR-filen. Dessa data återfinns på HDF5-form.

Denna visualisering ska även kunna göras för en godtycklig energi, inte bara för Fermi-energin. Denna energi väljs med hjälp av en slider eller genom att manuellt skriva in önskad energi.

Inviwo-nätverket för att rita upp Fermi-energin består av ett antal steg som beskrivs nedan. Se figur 15 för en skiss av hur dessa förhåller sig till varandra. Varje steg implementeras i Inviwo i form av en eller flera processorer.



Figur 15: Översiktlig beskrivning av Inviwo-nätverket för Fermi-yte-uppritning

- HDF5-inläsning
 - Hämtar data om Fermi-energin samt de möjliga energierna hos alla k-punkter inuti 1:a Brillouin-zonen
- Jämförelse

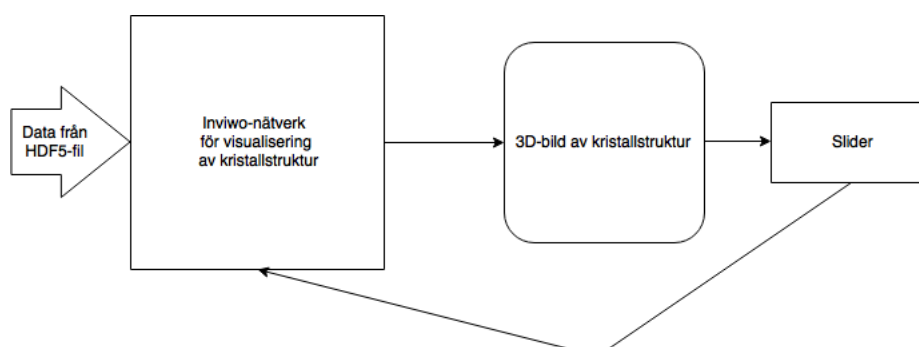
- Jämför de möjliga energierna för varje k-punkt med Fermi-energin för att se vilka k-punkter som ingår i Fermi-ytan
- Fermi-yte-konstruktion
 - Beräknar vilka k-punkter som utgör Fermi-ytan
- Canvas
 - Ritar upp Fermi-ytan

5.9 Dynamisk visualisering av kristallstruktur

Dynamisk visualisering baserad på en serie atompositioner. Användaren ska med hjälp av en slider kunna ändra visualiseringen.

5.9.1 Översiktlig beskrivning

Koordinaterna (tre dimensioner) för atomerna läses in från HDF5-fil. Utifrån dessa positioner ritas sedan atomerna ut. Användaren kan sedan med hjälp av en slider ändra tiden och se hur visualiseringen ändrar sig därefter. Se figur 16 för ett enklare flödesschema.



Figur 16: Översiktligt flödesdiagram över dynamik.

5.9.2 Inviwo-nätverk för dynamisk visualisering av kristallstruktur

En 3D-bild ska ritas ut med hjälp av ett antal steg. Det är denna 3D-bild som sedan kommer att uppdateras om användaren väljer att justera slidern för tid. Varje steg kommer att implementeras i Inviwo med en eller flera processorer.

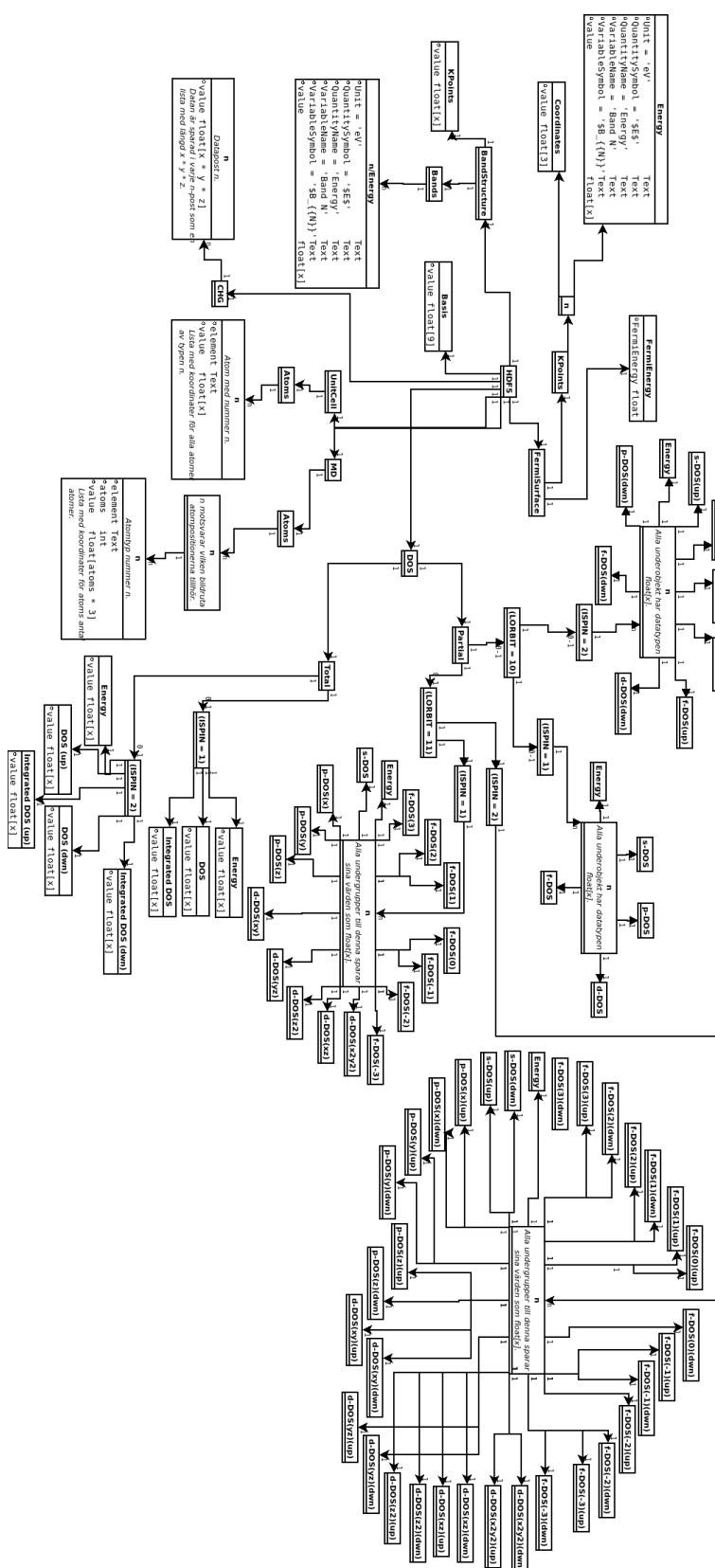
- HDF5-inläsning
 - Ladda in data från HDF5-fil.
- Koordinatläsning
 - Läs in koordinaterna för atomerna från HDF5-fil.
- Mesh-konstruering

- Bygga upp ett mesh utifrån koordinaterna från koordinatläsar-processorn.
- Canvas
 - Visa 3D-bilden.
- Slider
 - Ändra tiden.
- Koordinatläsning...

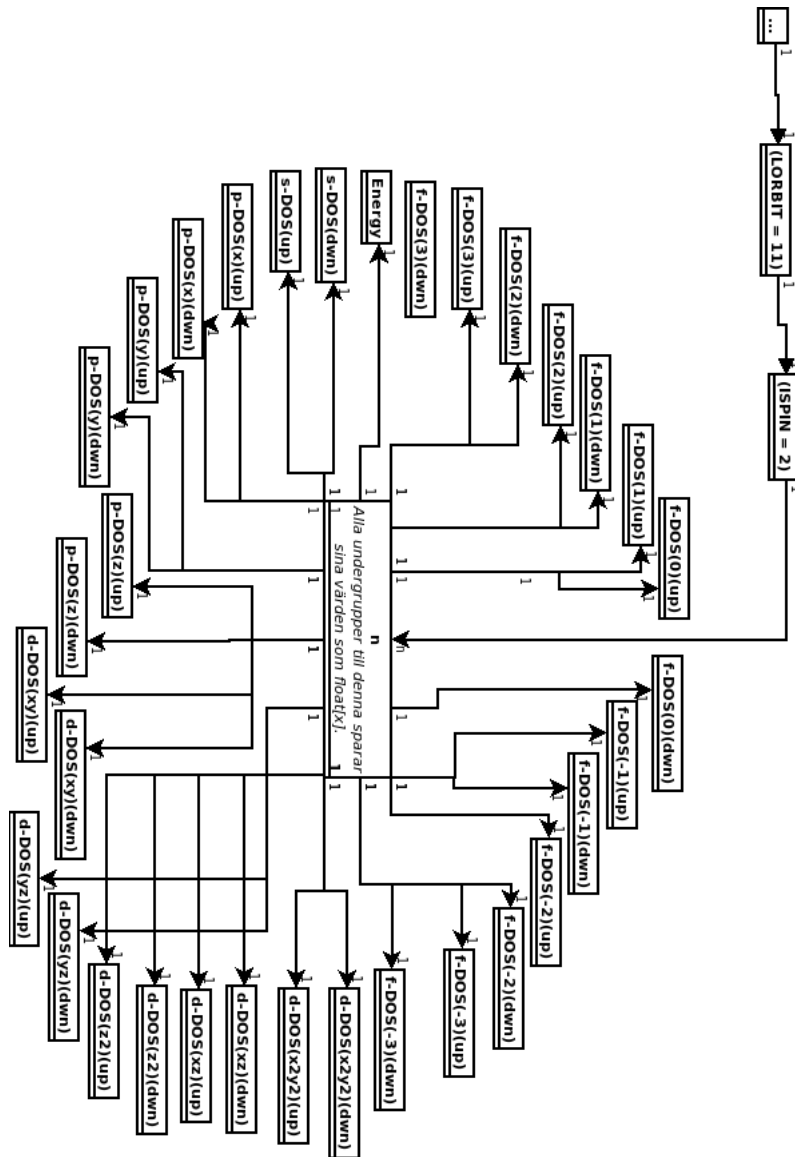
Referenser

- [1] *About VASP*. URL: <https://www.vasp.at/index.php/about-vasp/59-about-vasp> (hämtad 2018-02-22).
- [2] Neil Ashcroft och David Mermin. *Solid State Physics*. 1976, s. 141.
- [3] *BSD2*. URL: <https://opensource.org/licenses/BSD-2-Clause> (hämtad 2018-02-23).
- [4] *C++*. URL: <http://www.cplusplus.com/info/description/> (hämtad 2018-02-23).
- [5] *Git*. URL: <https://git-scm.com> (hämtad 2018-02-23).
- [6] The HDF Group. *Hierarchical Data Format, version 5*. 1997-2018. URL: <https://support.hdfgroup.org/HDF5/> (hämtad 2018-02-21).
- [7] The HDF Group. *High Level Introduction to HDF5*. 23 sept. 2016. URL: <https://support.hdfgroup.org/HDF5/Tutor/HDF5Intro.pdf> (hämtad 2018-02-21).
- [8] *Inviwo Release. HDF5 module*. URL: <https://www.inviwo.org/2017/10/04/inviwo-0-9-9-released/> (data) (hämtad 2018-02-27).
- [9] *Nationalencyklopedin. API*. URL: <https://www.ne.se/uppslagsverk/encyklopedi/l%C3%A5ng/api> (data) (hämtad 2018-02-23).
- [10] *Nationalencyklopedin. Fermi-yta*. URL: <https://www.ne.se/uppslagsverk/encyklopedi/l%C3%A5ng/fermi-yta> (hämtad 2018-02-23).
- [11] *Nationalencyklopedin. GUI*. URL: <https://www.ne.se/uppslagsverk/encyklopedi/l%C3%A5ng/api> (data) (hämtad 2018-02-23).
- [12] *Nationalencyklopedin. reciproka rymden*. URL: <http://www.ne.se/uppslagsverk/encyklopedi/l%C3%A5ng/reciproka-rymden> (hämtad 2018-02-27).
- [13] *Ordguru. Rastergrafik*. URL: <https://www.ordguru.se/synonymer/rastergrafik> (hämtad 2018-04-26).
- [14] *Python*. URL: <https://www.python.org> (hämtad 2018-02-23).

Figur 17: Dataformatet som används när VASP konverteras till HDF5.



Figur 17: Dataformatet som används när VASP konverteras till HDF5.



Figur 19: Dataformatet som används när VASP konverteras till HDF5 del 2.