Teknisk dokumentation

Redaktör: Anders Rehult

Version 0.1

Denna tekniska dokumentation är baserad på 2017 års tekniska dokumentation för visualiseringsvertyget ENVISIoN.

Status

Granskad	PL	18-05-18
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

Version	Datum	Utförda förändringar	Utförda av	Granskad
0.1	2018-05-18	Första utkast.	Projektgruppen	

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 Inviwo och programmeringspråken Python och C++ användas. Det kommer inte utredas om det är bättre att använda andra verktyg.

1.5 Definitioner

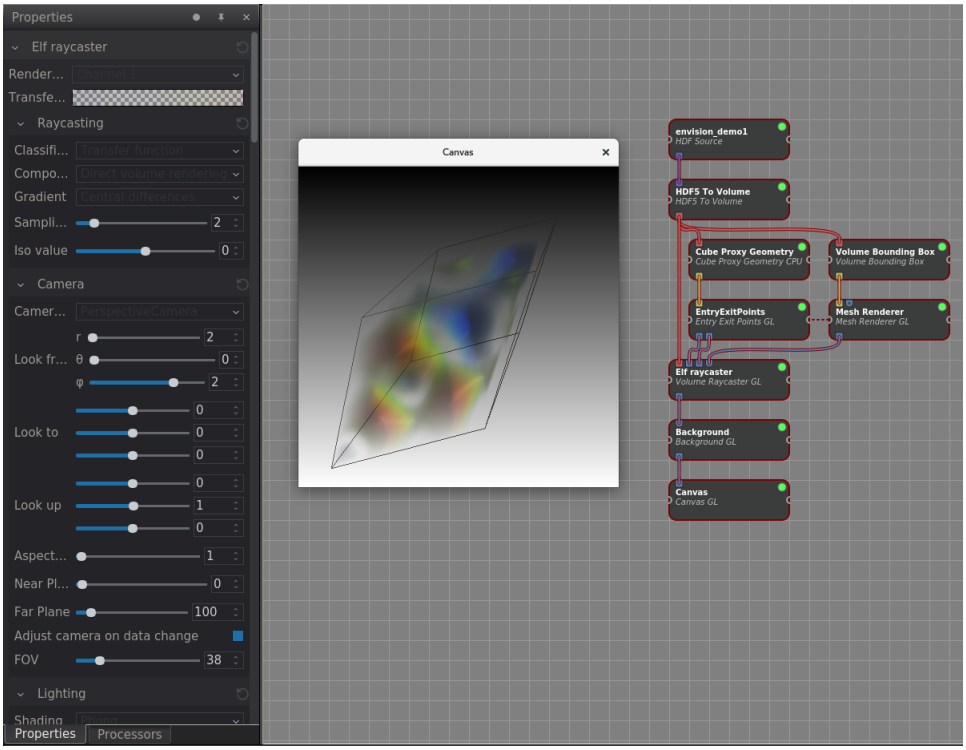
- **Inviwo:** Ett forskningsverktyg som utvecklas vid Linköpings universitet och ger användaren möjlighet att styra visualisering med hjälp av programmering i Python 3 eller grafiskt. Det tillhandahåller även användargränssnitt för interaktiv visualisering.
- **Processor:** Benämningen på ett funktionsblock 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. [8]
- **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. [12]
  I Inviwo används Python för att knyta samman processorer.
- **Fermi-energi** defineras 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 rymden, isoytan där elektronernas energi är lika med Fermi-energin. [9]
- **Git** är ett decentraliserat versionshanteringssystem. [5]
- **GUI** (Graphical User Interface) är ett grafiskt användargränssnitt. [10]
- **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 gås inte igenom i detalj i detta dokument, men finns väl beskrivna i *High Level Introduction to HDF5* [7].
- **VASP** är ett program för modellering på atomnivå, för t.ex. elektronstruktusrberäkningar och kvantmekanisk molekyldynamik. [1]
- **Rastergrafik** är en typ av data som innehåller en matris av färgvärden, med bestämd och informationsbegränsad upplösning. [11]

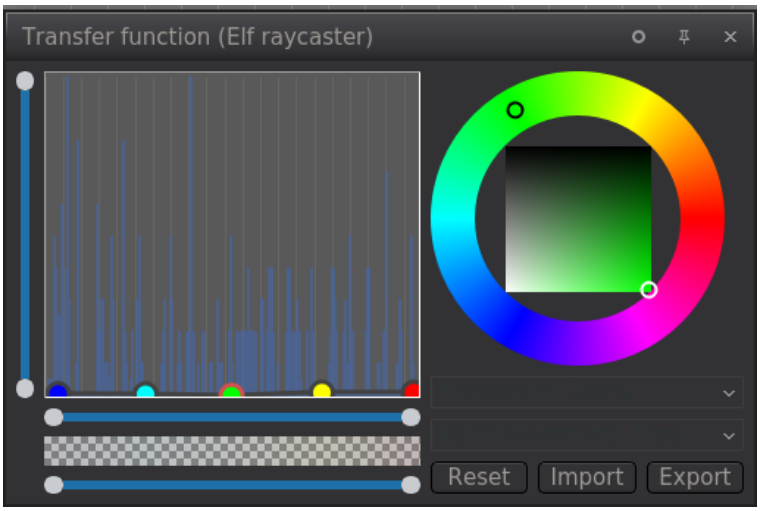
2 Produkten

Visualiseringsverktyget kan visualisera elektrontäthet, ELF (Electron localization function), kristallstruktur som atompositioner i enehetscellen samt total och partiell tillståndstäthet.

Figur 1 visar vad användaren ser när ELF-data för diamant visualiseras. Längst till vänster i figuren ses ett fönster med Properties, i detta fall för processorn ELF raycasting som kan hittas i nätverket längst till höger i figuren. Properties för de olika processorerna fås när en processor är markerad. I mitten av figuren ligger den utritade bilden. Bilden kan roteras genom att klicka och dra i den. I fönstret Properties finns en Property vid namn Transfer function, med hjälp av vilken användaren kan ändra transparens/opacitet samt färg, se figur 2.



Figur 1: Bild av vad användaren ser när ELF visualiseras, i detta fall för diamant.



Figur 2: Transfer function property där transparens och färg kan ställlas in.

3 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.

3.1 Resurser

Följande resurser har använts för att utveckla systemet:

Mjukvara:

- Inviwo
- h5py
- Qt Creator
- CMake
- Git

Programmeringsspråk:

- Python 3
- C++ 14

3.2 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.

3.3 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.

4 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.

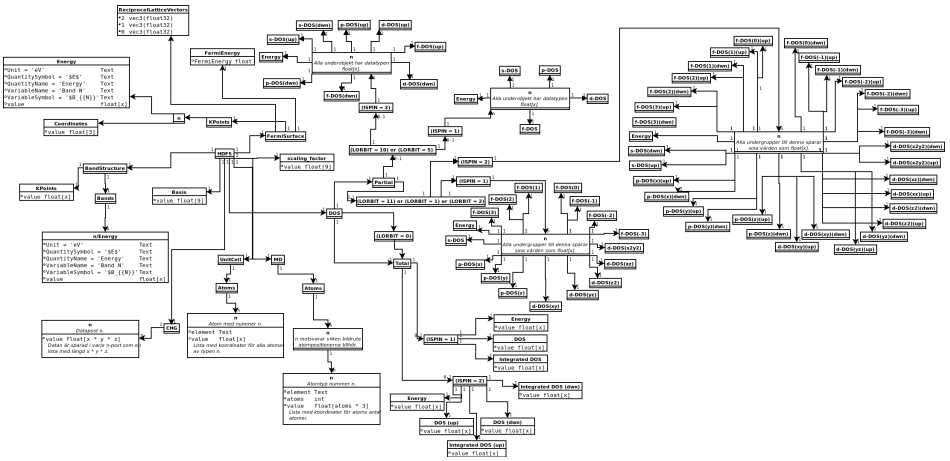
4.1 VASP

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 4.4.3 för mer information.
- EIGENVAL innehåller data för alla energier i k-rummet.
- POTCAR innehåller data om atomtyper.
- OUTCAR innehåller all utdata.
- 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.

4.2 HDF5-struktur för VASP-data

För att enkelt kunna läsa data i Inviwo följer HDF5-filerna som skapas av parsersystemet filstruktur enligt figur 3 nedan. Figur 3 beskriver det HDF5-format som VASP-datan ska läsas in till.



Figur 3: Dataformatet som används när VASP konverteras till HDF5. Se större bild i Bilaga A och B.

En ruta i diagrammet motsvarar en del i sökvägen i HDF5-objektet, om inte titeln är inom paranteser, 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 3 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*.

4.3 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 3 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: Skalär 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.

4.4 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.

4.4.1 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 4.2 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.

4.4.2 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.

4.4.3 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.

Parametrar:

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

Returnerar:

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

4.4.4 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.

4.4.5 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.

5 Delsystem 2 - System för visualisering

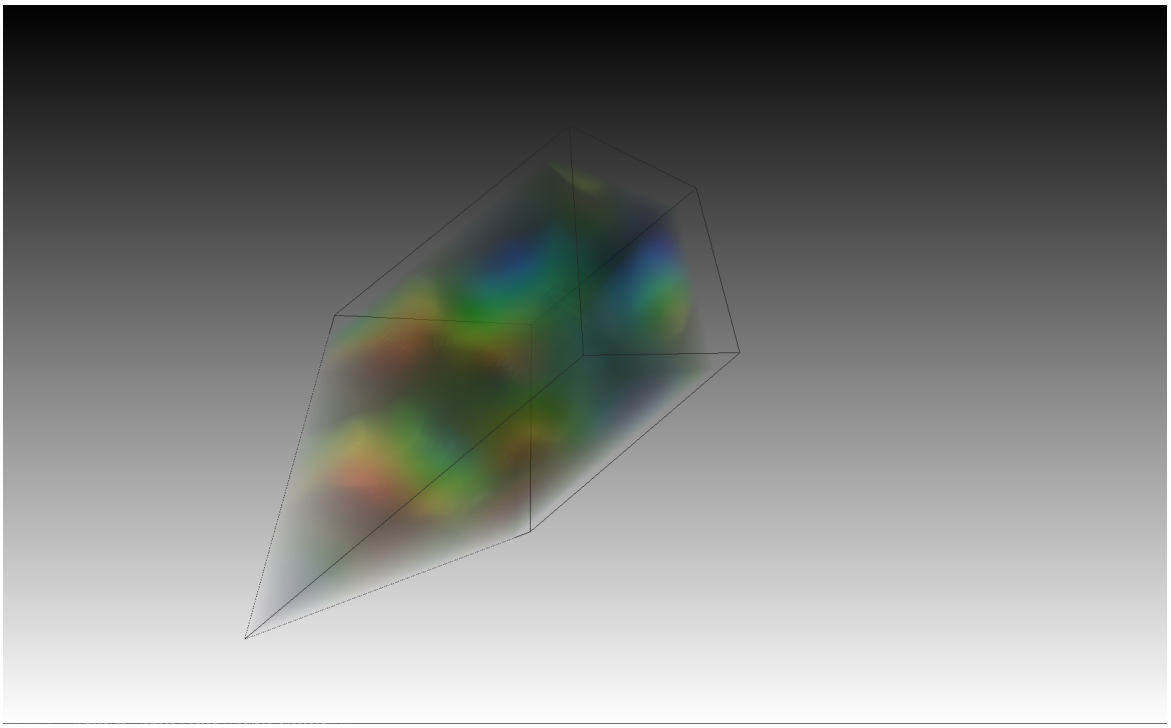
Visualiseringssystemet består av ett antal pythonmoduler för nätverksbygge i Inviwo samt c++-moduler implementerade i Inviwo av projektgruppen för att visualisera data.

5.1 Nätverk

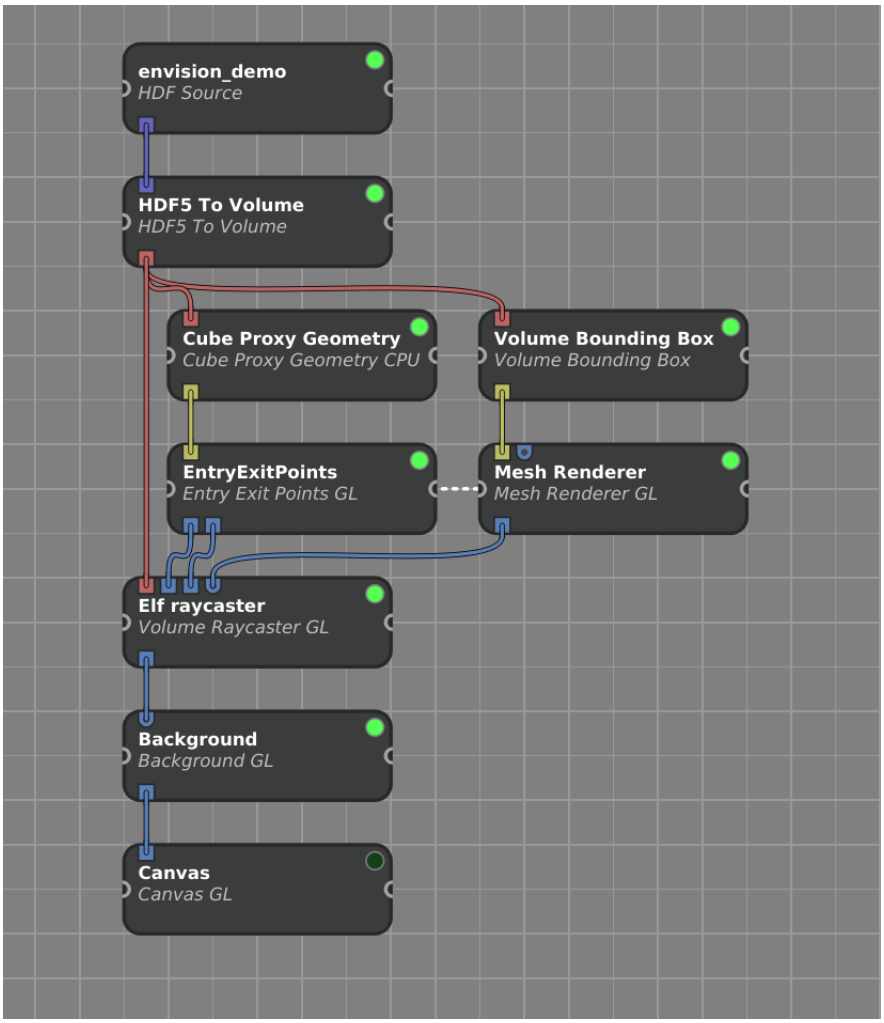
Processorer är de primära objekt i nätverken som användaren interagerar med. De består bland annat av inportar och utportar som används vid utbyte av data av specifika typer. Nätverken mynnar ut i en så kallad Canvas, en processor som på sin inport tar rastergrafikdata som den visualiserar i ett separat 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.

5.1.1 Nätverk för visualisering av volymdata

Data som visualiseras med hjälp av volymnätverket är elektrontäthetsdata och elektronlokaliseringsfunktionsdata (ELF-data). Nätverket kan visualisera data med hjälp av tekniken volume ray casting eller iso ray casting. Se figur 4 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 5.

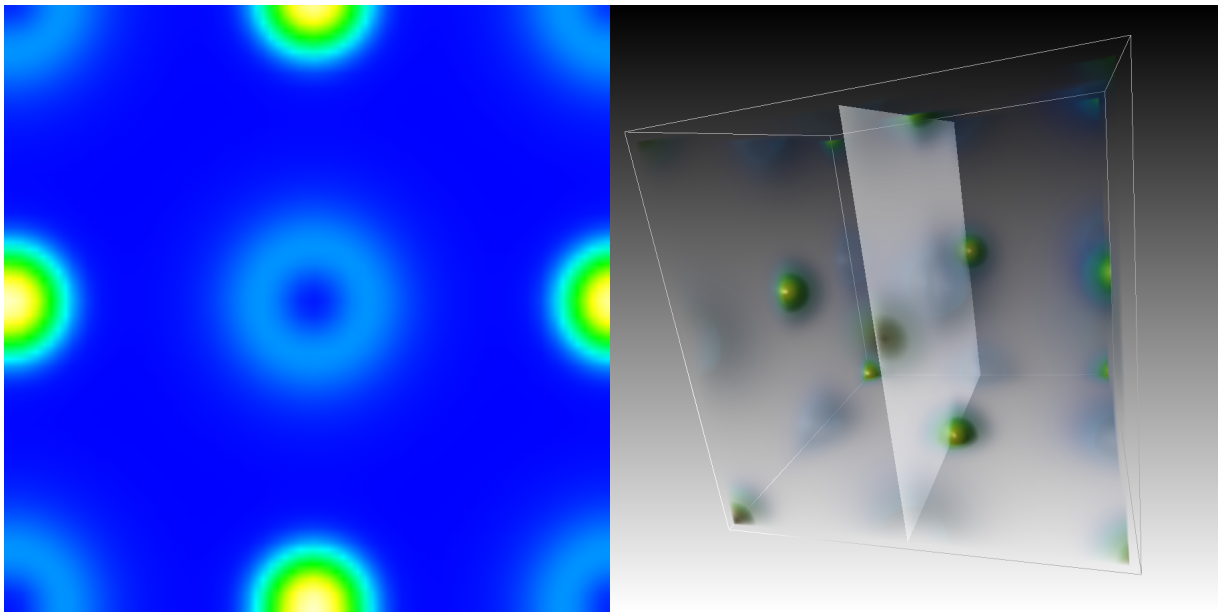


Figur 4: Visualisering av ELF-data för diamant i Inviwo.

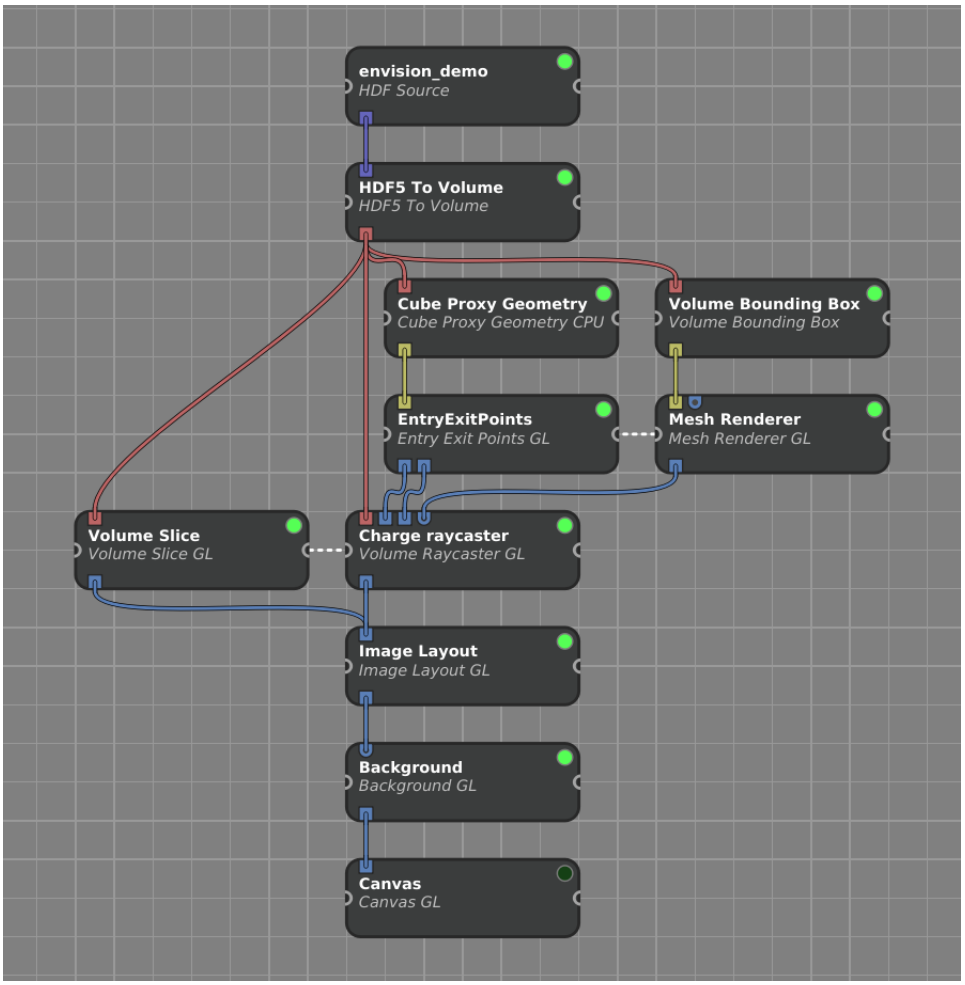


Figur 5: Nätverket för visualisering av ELF-data för diamant i Inviwo

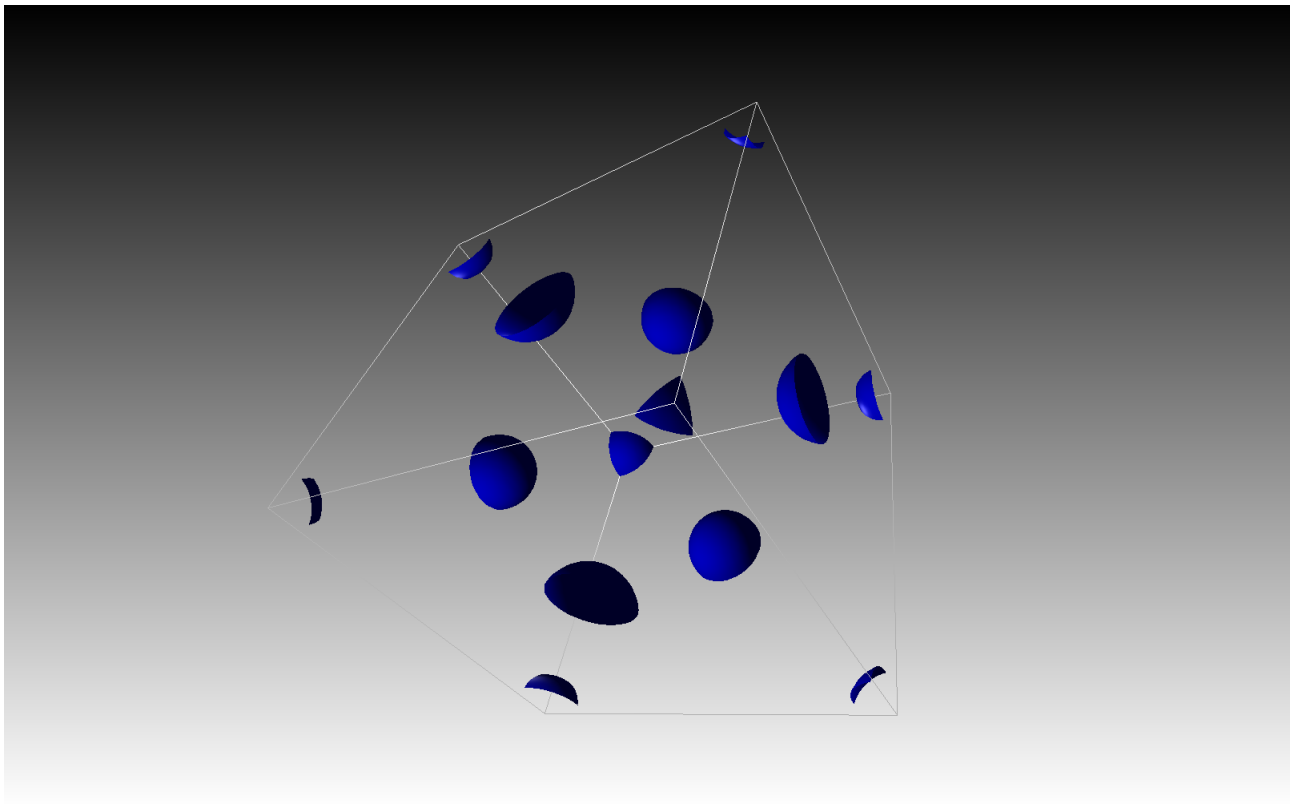
När tekniken volume ray casting används har användaren möjlighet att aktivera en slice-funktion som skapar ett plan som skär igenom volymen. Visualiseringen består då av två stycken bilder placerade bredvid varandra i samma Canvas. Högra bilden visar hela volymen tillsammans med ett plan, som användaren 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 6 nedan som visar en skärmbild från Inviwo när visualisering av laddningstäthet för natriumklorid, NaCl, med volume ray casting och slice-funktion görs, och figur 7 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 8 visar visualisering av laddningstäthet för NaCl med ISO ray casting, där en isoyta ritats upp för laddningstäthet lika med 0.15.



Figur 6: Visualisering av laddningstäthet för NaCl med slice-funktion i Inviwo.



Figur 7: Nätverket för visualisering av laddningstäthet för NaCl med slice-funktion i Inviwo.



Figur 8: Visualisering av laddningstäthet för NaCl med ISO raycasting.

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.
- ytart_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.

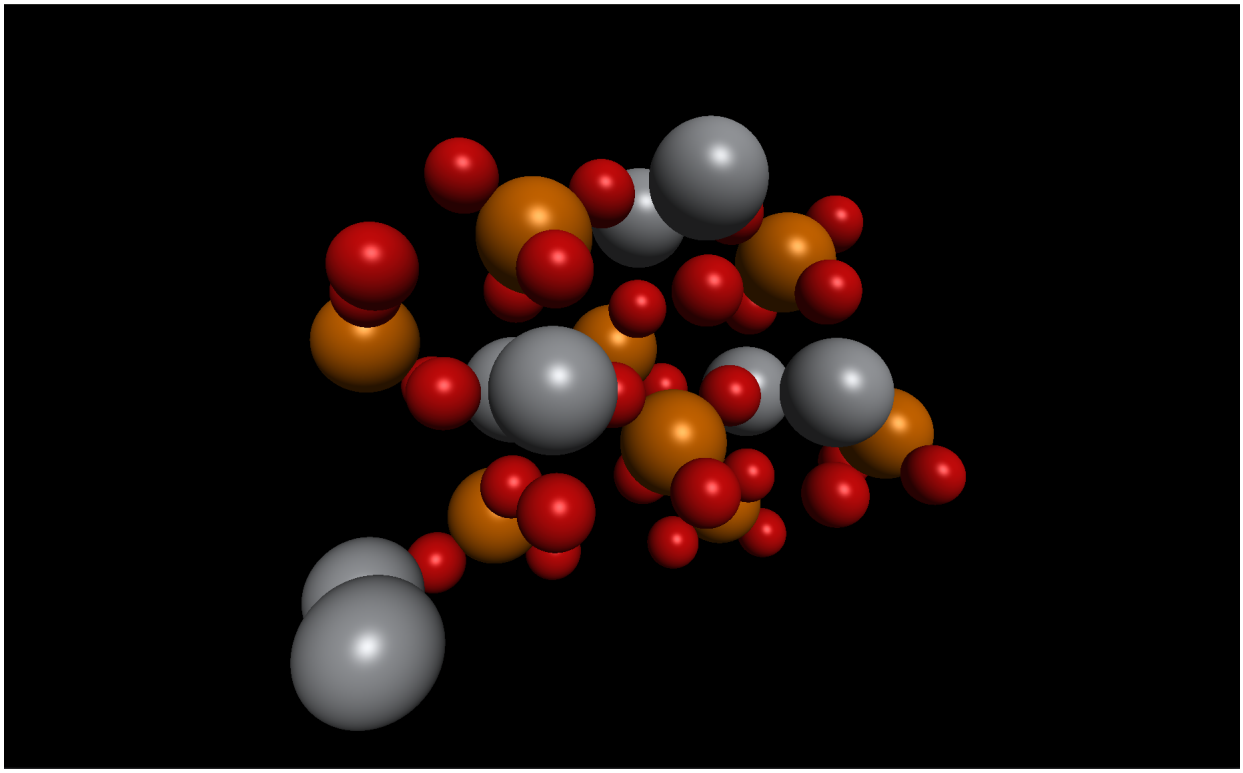
Funktionen volume_network laddar en HDF Source processor som ges samma namn som HDF5-filen. Dess utdata skickas till en HDF5 To Volume processor. HDF Source processorn anger vilken HDF5-fil och HDF5 To Volume processorn anger, baserat på argumentet “volume”, vilken specifik grupp i HDF5-filen som data 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 parallellepiped 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 5 som visar en skärmbild från Inviwo över nätverket när ELF-data för diamant visualiseras med volume ray casting.

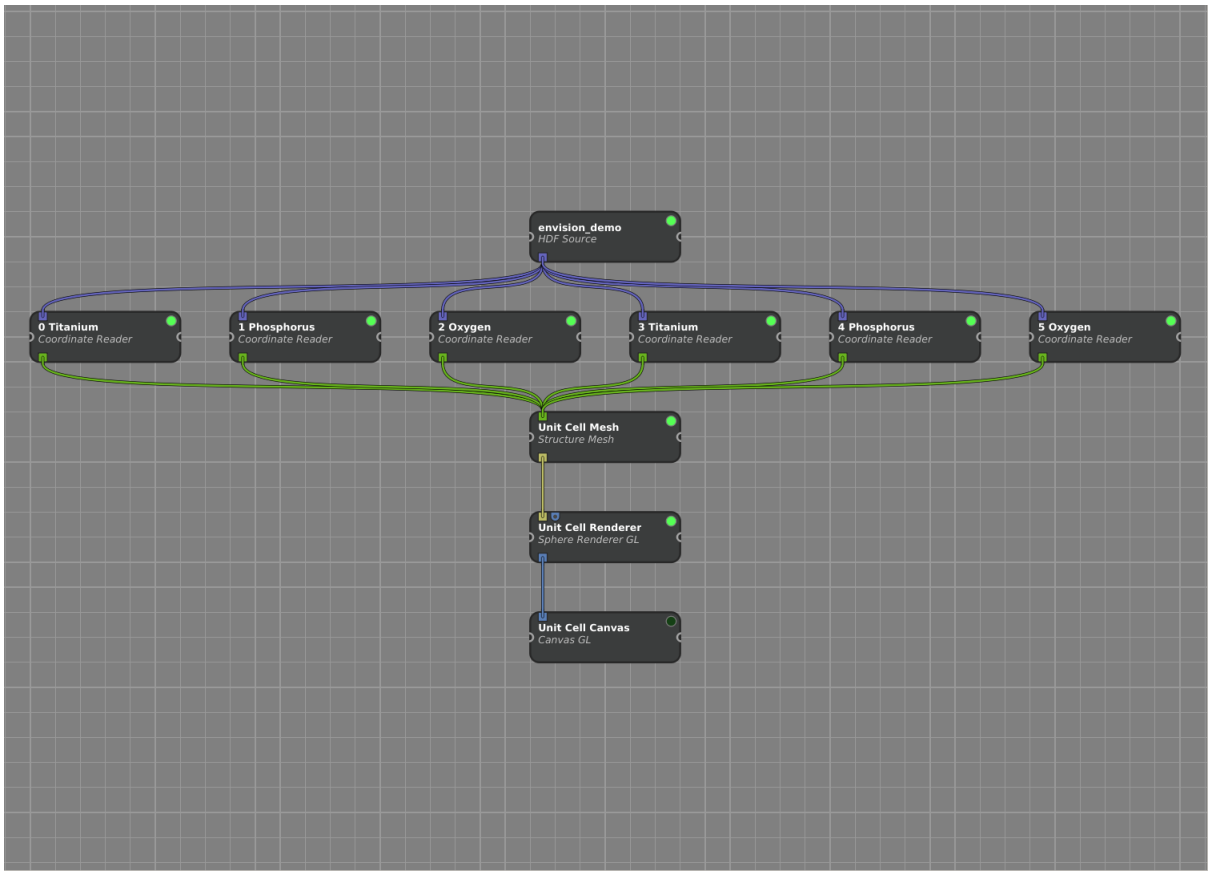
Om slicefunktionen används laddar även funktionen volume_network en Volume Slice processor och en Image Layout processor. Volume Slice processorn skapar det plan som skär volymen se figur 6 till höger. Indata till denna processor utgörs av utdata från HDF5 To Volume processorn och dess utdata går till processorn Image Layout. Processorn Image Layout skapar en delad vy över de båda bilderna se figur 6 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 7 som visar en skärmbild från Inviwo över nätverket när laddningstäthet för natriumklorid, NaCl, visualiseras med hjälp av volume ray casting och slice-funktion.

5.1.2 Nätverk för visualisering av enhetscell

Funktionen för visualisering av enhetscell placerar ut en koordinatläsarprocessor (CoordinateReader) 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 StructureMeshprocessor 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 5.1.1 om detta nätverk ovan, så att enhetscellen visas tillsammans med elektronstrukturen. Se figur 9 nedan som vardera visar en skärmbild från Inviwo när visualisering av enhetscell för titanfosfat (TiPO4) görs, och figur 10 för tillhörande nätverk.



Figur 9: Visualisering av TiPO4 i Inviwo

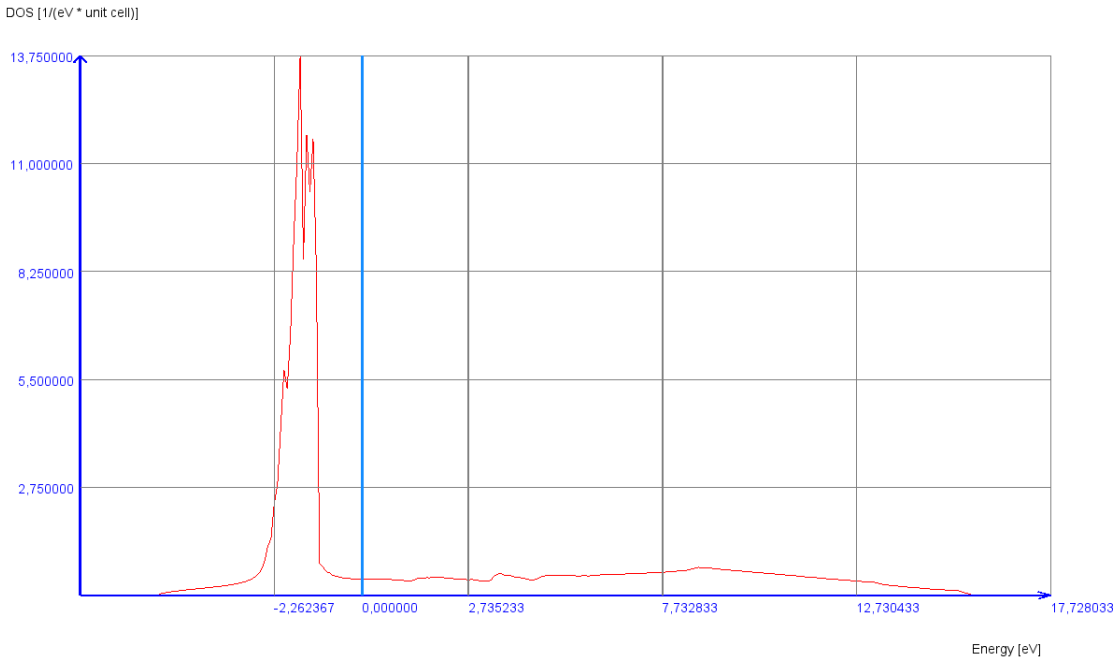


Figur 10: Nätverket för visualisering av TiPO4 i Inviwo.

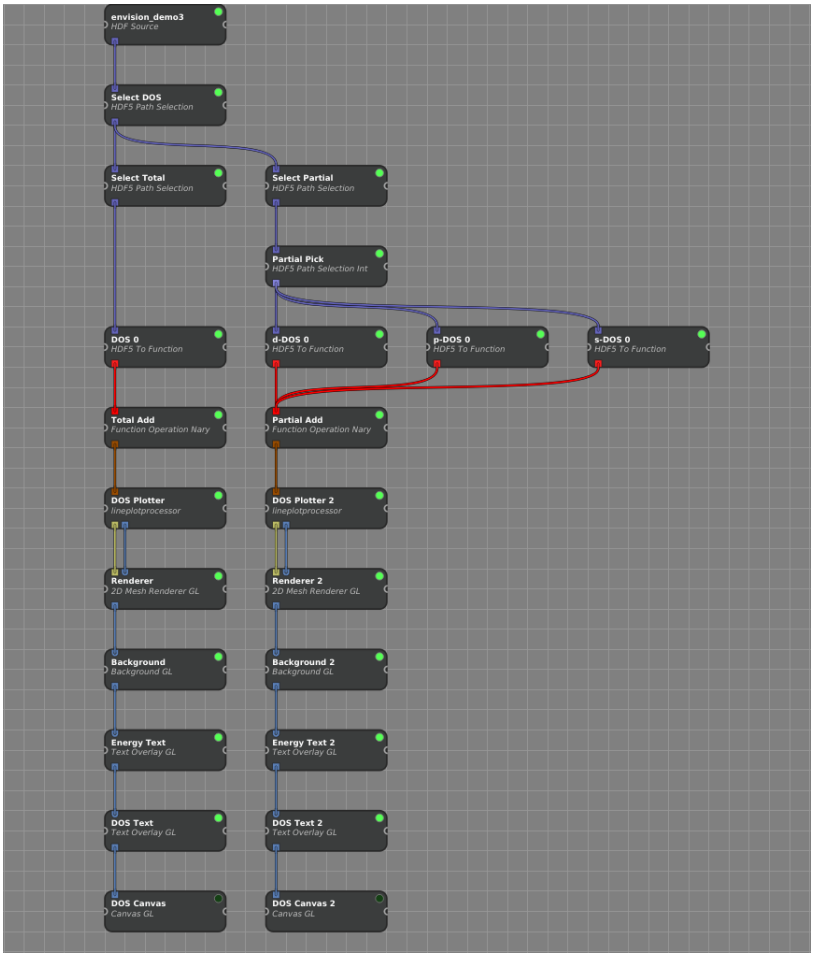
5.1.3 Nätverk för visualisering av DOS

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, som tar ut den givna HDF5-gruppens alla undergrupper (beskrivs närmare i kap. 5.3) 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 HDF5 To Function processorer som gör funktioner av all data i Total-gruppen. För Partial-gruppen laddas en HDF5 Path Selection processor (beskrivs närmare i kap. 5.3.2) som tar ut dataset för en vald atom genom att välja den givna HDF5-filens relevanta undergrupp. Denna processor har namnet Partial Pick i nätverket. Därefter laddas HDF5 To Function processorer för alla dataset i grupperna under Partial-gruppen. All data matas sedan i en lineplot processor (beskrivs närmare i kap. 5.3.3) som gör en 2D-graf. Detta matas in i en Canvas-processor som visar själva grafen. Dessutom finns två textOverlay processorer som skriver ut text för x- och y-axeln. Figur 11 visar total tillståndstäthet för koppar, Cu. Skärmbilden i figur 12 är över nätverket som ger 2D-grafen i figur 11, nätverket ger även en 2D-graf av den partiella tillståndstätheten.



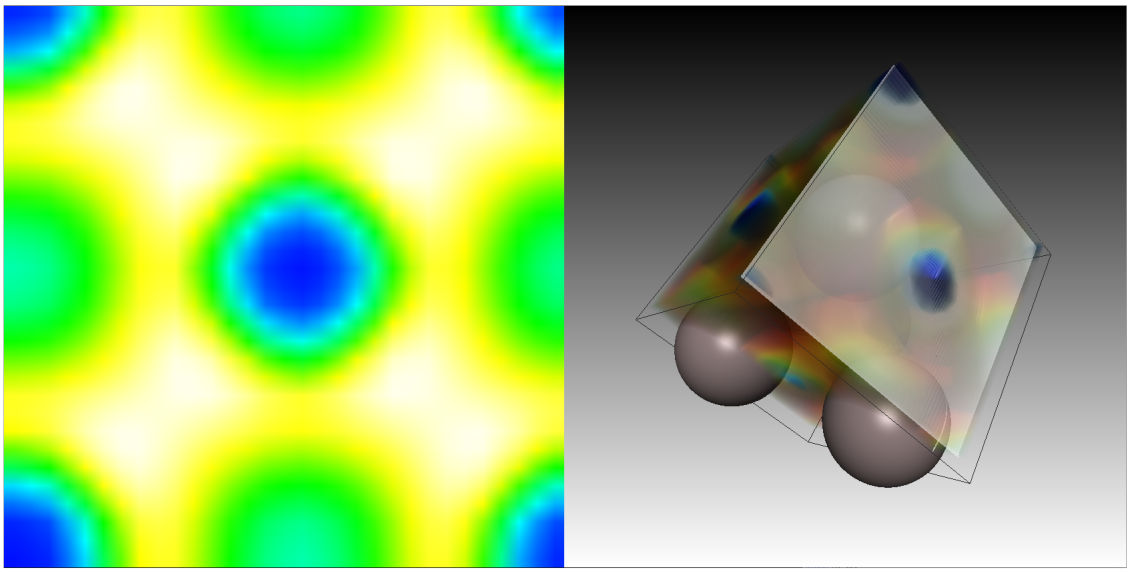
Figur 11: Visualisering av Total DOS för Cu.



Figur 12: Nätverket för visualisering av total och partiell tillståndstäthet för Cu.

5.1.4 Sammankoppling av nätverk

Alla de typer av visualisering som är listade ovan kan köras samtidigt, oberoende av varandra, så länge nödvändig data återfinns i de tillhandahållna VASP-filerna. Funktionalitet för att koppla ihop nätverk med varandra och visualisera flera egenskaper i samma bild finns för några kombinationer av egenskaper. Alla kombinationer av tredimensionell visualisering i det "direkta" rummet (alltså inte det reciproka rummet) kan fås i samma bild. Ett exempel på detta är figur 13 som visar visualisering av enhetscellsdata och ELF-data för aluminium, Al, i samma bild.



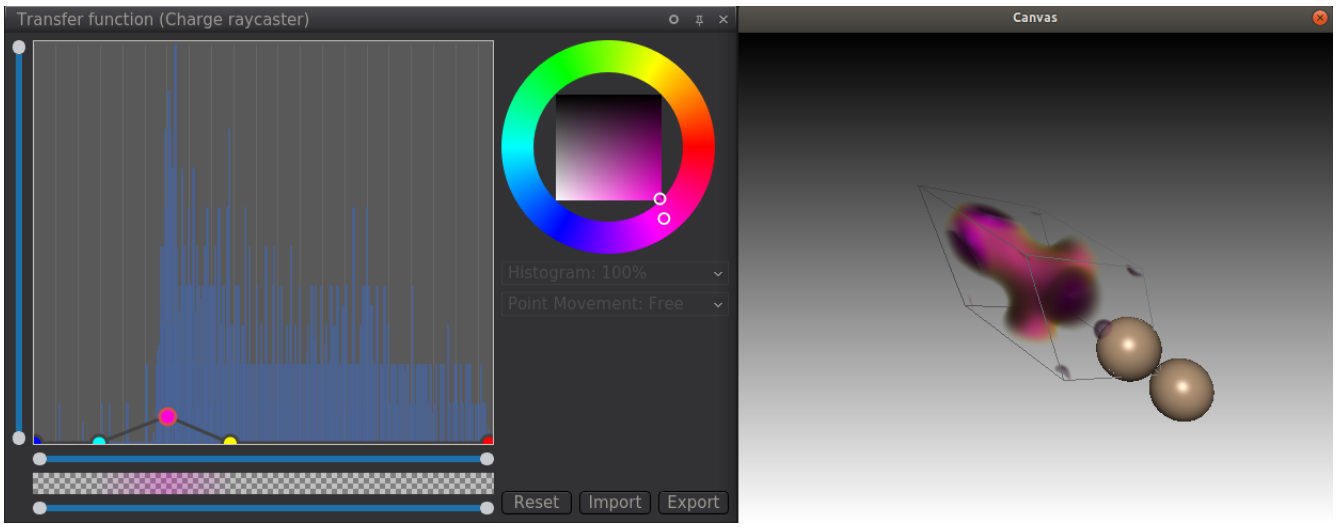
Figur 13: Data för enhetscell och ELF för aluminium visualiserat i samma bild, med volume ray casting och slice-funktion. Slice-funktionens plan är här placerat alldeles intill enhetscellens sida.

5.1.5 Färg-och-transparensinställning

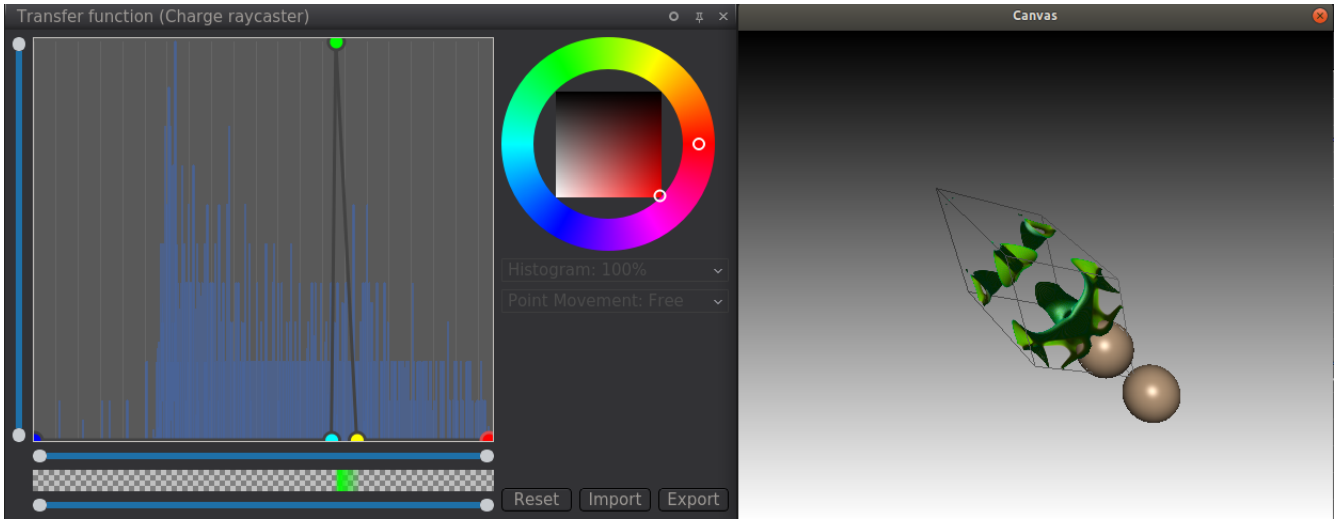
Färger och transparens för samtliga typer av tredimensionell visualisering kan ändras godtyckligt av användaren.

- Enhetscellsvisualisering: Färg och transparens ändras i processorn *Unit Cell Mesh* i propertyn *color i*, där *i* är numret på den atomtyp som inställningarna ska ändras för.
- Visualisering av volymdata: Färg och transparens (ej för ISO raycasting, då isoytor ritas helt opaka) ändras i processorn *Charge Raycaster* eller *ELF Raycaster* för laddningstäthetsvisualisering respektive ELF-visualisering, i propertyn *Transfer function*. Här visas en interaktiv graf med volymsdatavärden längs x-axeln och opacitet längs y-axeln. Med

hjälp av punkter som placeras ut i detta plan kan användaren styra färgen och transparensen på vad som visualiseras. Se figur 14 och 15 för exempel på dessa inställningar för överföringsfunktionen för visualiseringen av laddningstäthet av kisel i diamantstruktur.



Figur 14: Laddningstäthetsvärden mellan 0.15 och 0.45 markeras med en någorlunda genomskinlig lila yta.



Figur 15: Laddningstäthetsvärden kring 0.7 markeras med en opak grön yta, nästan en isoyta.

Slice-funktioners plans färg och transparens kan även de ändras. Detta i processorn *Volume Slice*, i propertyn *Transfer function*.

5.2 Datastrukturer

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

5.2.1 Point

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

5.2.2 Function

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

5.3 Processorer

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

5.3.1 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 flyttal. Utdata från CoordinateReader är n stycken vec3.

Inport:

- Hdf5::Inport inport_

Utport:

- DataOutport< std::vector<vec3> > outport_

Properties:

- StringProperty path_

StructureMesh
Atompostionsdata kopplas ihop med rätt atomfärg och radie med StructureMesh-processorn. Dessutom hanterar processorn ”picking”-funktionen som gör det möjligt för användaren att välja atomer genom att klicka på dem. StructureMesh har en multiinport, 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.

Om picking-funktionen är påslagen får meshen även en pickingbuffer, som innehåller de globala picking id som tilldelas av PickingMappern. Då användaren vänsterklickar på en atom läggs dess lokala id i IntVectorPropertyn. Färgen ändras på alla valda atomer genom att alfalagret sätts till 0,5.

Inport:

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

Utport:

- MeshOutport mesh_

Properties:

- FloatProperty scalingFactor_
- FloatMat3Property basis_
- BoolProperty fullMesh_
- 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
- BoolProperty enablePicking_: sann då picking-funktionen är påslagen
- IntVectorProperty inds_: vektor med index på valda atomer

5.3.2 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 utökat 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 mönster som, och fungera väl med, resterande processorer.

Inport:

- DataInport<hdf5::Handle> hdf5HandleInport_

Utport:

- DataOutport< std::vector<hdf5::Handle> > hdf5HandleVectorOutport_

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_.

HDF5PathSelectionIntVector kan med fördel användas tillsammans med ”picking” för att plocka ut relevant data ur en HDF5-fil.

Inport:

- DataInport<hdf5::Handle> hdf5HandleInport_

Utport:

- DataOutport< std::vector<hdf5::Handle> > hdf5HandleVectorOutport_

Properties:

- IntVectorProperty intVectorProperty_
- IntSizeTProperty 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 mellan liggande 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_, metadat utökas med namnen på de grupper var i 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_
- IntSizeTProperty 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:

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

Utport:

- DataOutport< std::vector<Function> > functionVectorOutport_

Properties:

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

5.3.3 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 ($g_i(x) = 1/f_i(x)$). Operatorn appliceras på funktioner på inporten, en i taget, och skriver respektive resultat på utporten.

Inport:

- DataFrameInport dataframeInport_

Utport:

- DataFramOutport dataframOutport_

Properties:

- OptionPropertyString operationProperty_

FunctionOperationNary
Denna processor implementerar en operator med variabel aritet (engelska n-ary), antingen addition/summa ($g(x) = \Sigma_i f_i(x)$) eller multiplikation/produkt ($g(x) = \Pi_i f_i(x)$). Operatorn appliceras på samtliga funktioner på inporten och skrver 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.

Inport

- org.envision.FunctionFlatMultiInport functionFlatMultiInport_

Utport:

- DataFramOutport dataframOutport_

Properties:

- OptionsPropertyString operationProperty_

- OptionsPropertyString undefinedFallbackProperty_
- BoolProperty sampleFilterEnableProperty_
- FloatProperty sampleFilterEpsilonProperty_

lineplotprocessor
lineplotprocessor tar en *DataFrame* som förväntas innehålla två kolumner med punkter, kallade X och Y. Den konstruerar en mesh som representerar en linjegraf och denna mesh renderas sedan förslagsvis med hjälp av en *2D Mesh Renderer*-processor för att generera en bild av grafen.

lineplotprocessor genererar även en utbild att lägga över grafen som innehåller axelgraderingen. Axelgraderingen kan också den skickas in i *2D Mesh Renderer*-processorn och kommer då läggas ovanpå grafen.

Inställningar som har *range* i namnet justerar minimum- och maximumvärden på koordinataxlarna. Inställningar med *width* eller *colour* justerar bredd respektive färg för olika linjer ritade i diagrammet.

*label_number_* anger antalet divisioner på koordinataxlarna. Är värdet till exempel satt till tjugo innebär det att varje axel kommer ha tjugo divisioner och tjugo axelgraderingsetiketter.

*font_* ställer in vilket typsnitt axelgraderingen skall ha.

*enable_line_* aktiverar ritandet av en vertikal linje på x-koordinaten specificerad i *line_x_coordinate_*. Denna är avsedd att ge en visuell markering av var specifika x-värden finns på x-axeln.

Inport:

- DataFrameInport dataFrameInport_

Utports:

- MeshOutport meshOutport_
- ImageOutport labels_

Properties:

- FloatVec4Property colour_
- FloatVec2Property x_range_
- FloatVec2Property y_range_
- FloatProperty scale_
- BoolProperty enable_line_
- FloatProperty line_x_coordinate_
- FloatVec4Property line_colour_
- FloatVec4Property axis_colour_

- FloatProperty axis_width_
- FloatVec4Property grid_colour_
- FloatProperty grid_width_
- FontProperty font_
- FloatVec4Property text_colour_
- IntProperty label_number_

5.4 Properties och widgets

5.4.1 IntVectorpropety

Denna property består av en vektor av int-värden.

5.4.2 IntVectorPropertyWidget

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

6 Utvecklingsmöjligheter

De egenskaper som visualiserats under 2018 års projekt kan fortfarande vidareutvecklas för att få nya funktioner eller för att presentera datan på ett annat sätt om det anses mer lättförståeligt eller givande.

Molekyldynamik och bandstruktur är två egenskaper som inte har behandlats under detta års projektarbete men som det finns möjlighet att uppdatera från 2017 års projektarbete. I StructureMesh-processorn kan en property läggas till som möjliggör molekyldynamik. För bandstrukturvisualisering finns en parser och samt en visualiseringsmodul som kräver uppdatering.

Detta år har Fermi-ytor varit en av egenskaperna som skulle visualiseras, det hann dock inte bli helt klart. Mycket arbete har ändå lagts ner som det finns möjlighet att bygga vidare på och få att fungera.

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] *Nationalencyklopedin. API*. URL: https://www.ne.se/uppslagsverk/encyklopedi/l%C3%A5ng/api-(data) (hämtad 2018-02-23).

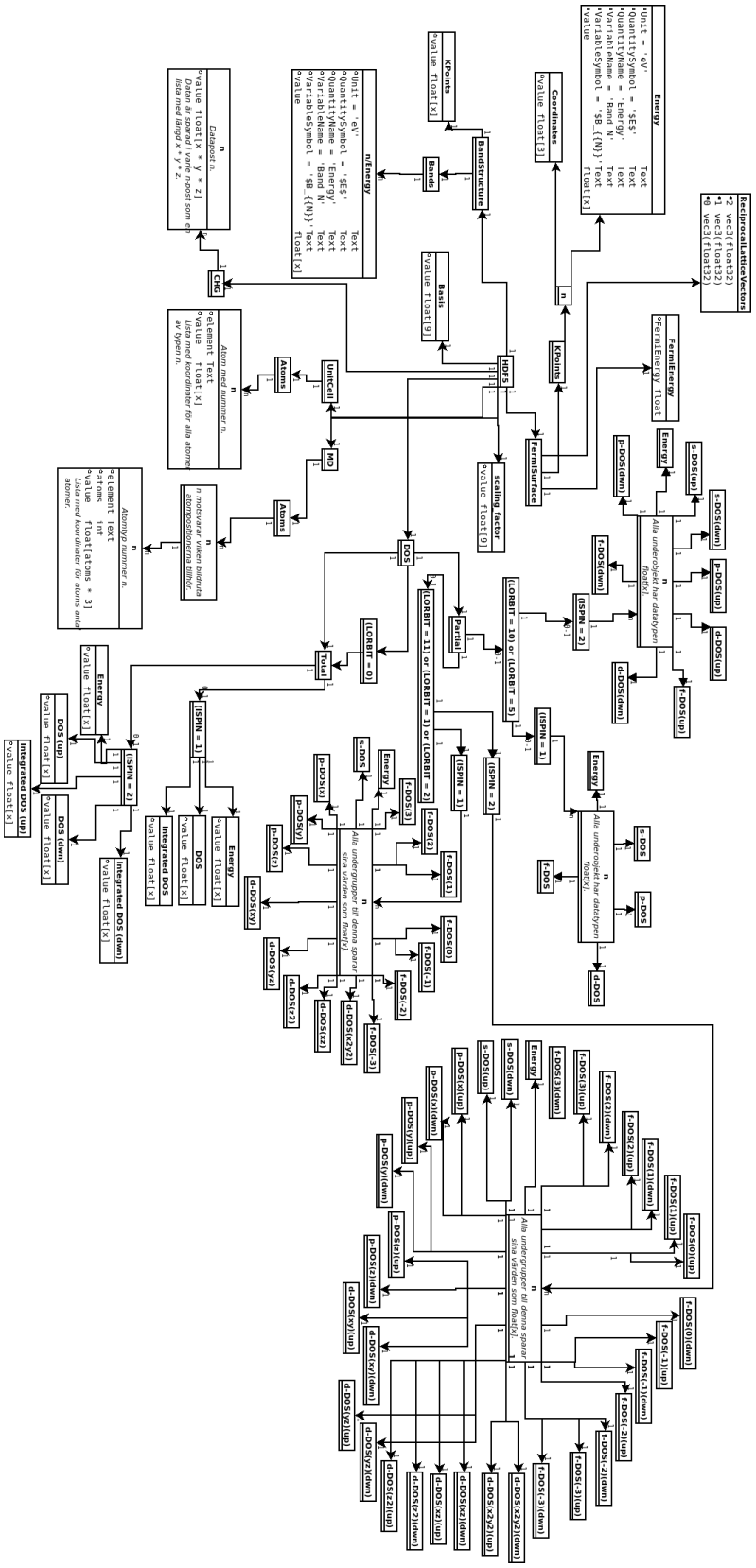
[9] *Nationalencyklopedin. Fermi-yta*. URL: https://www.ne.se/uppslagsverk/encyklopedi/l%C3%A5ng/fermi-yta (hämtad 2018-02-23).

[10] *Nationalencyklopedin. GUI*. URL: https://www.ne.se/uppslagsverk/encyklopedi/l%C3%A5ng/api-(data) (hämtad 2018-02-23).

[11] *Ordguru. Rastergrafik*. URL: https://www.ordguru.se/synonymer/rastergrafik (hämtad 2018-04-26).

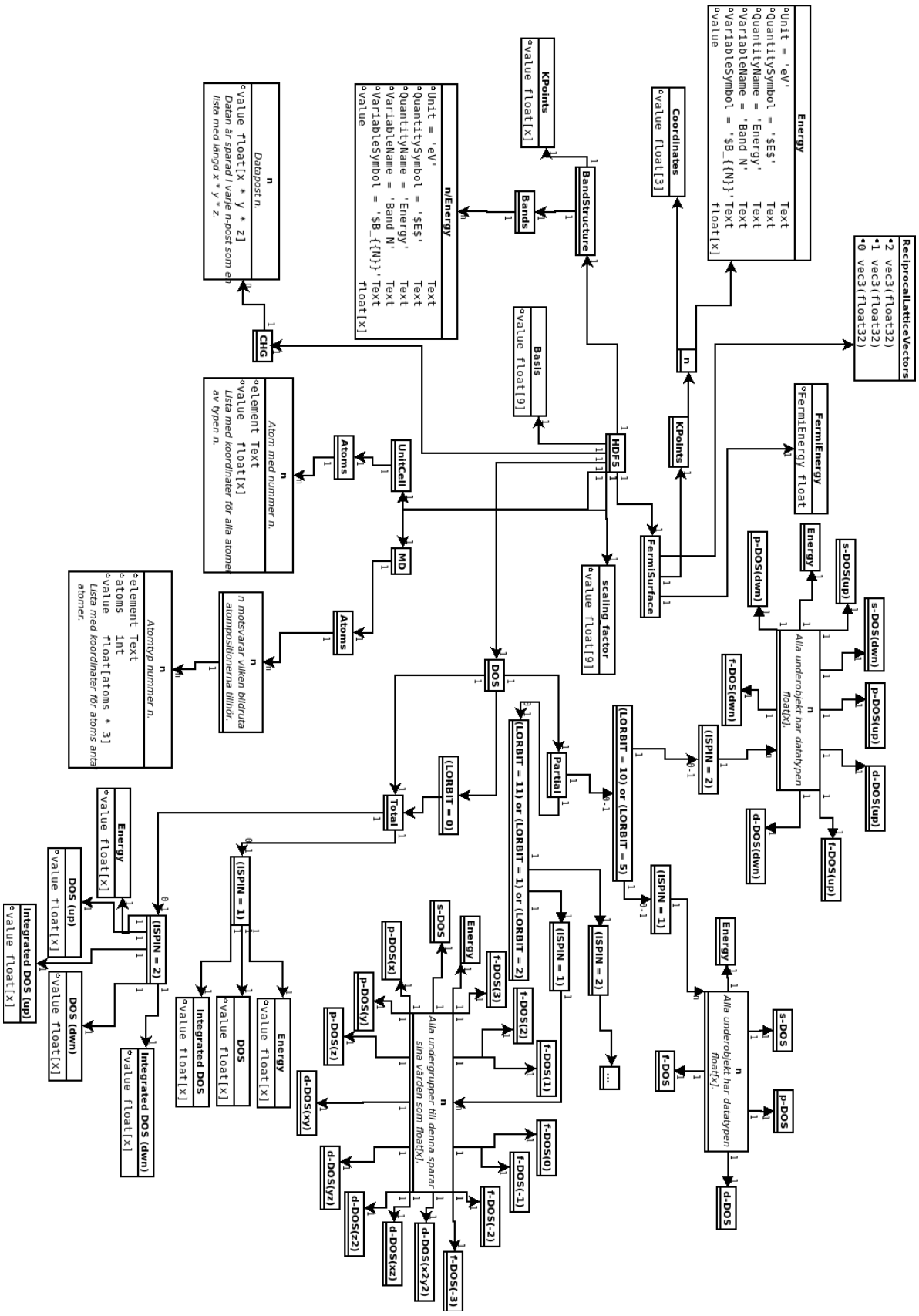
[12] *Python*. URL: https://www.python.org (hämtad 2018-02-23).

A HDF5-datastruktur

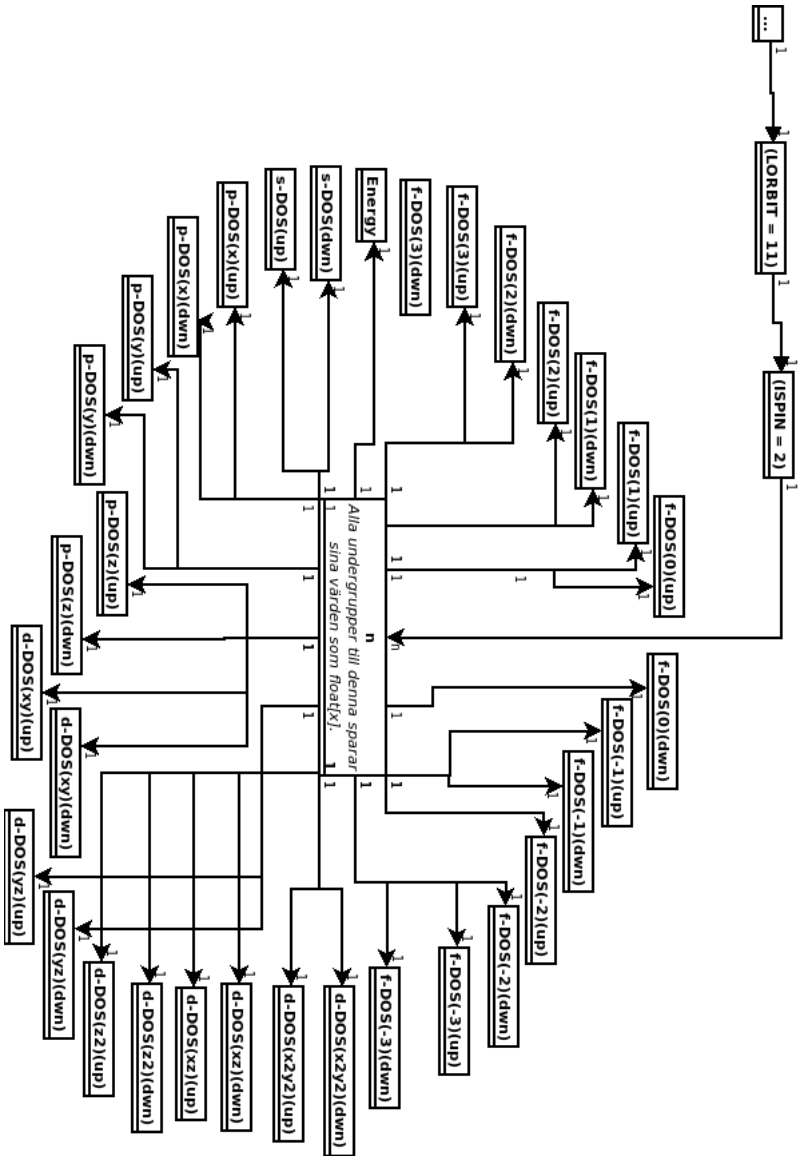


Figur 16: Dataformatet som används när VASP konverteras till HDF5.

B HDF5-datastruktur i två delar



Figur 17: Dataformatet som används när VASP konverteras till HDF5 del 1.



Figur 18: Dataformatet som används när VASP konverteras till HDF5 del 2.