Výzkumná zpráva:
Návrh architektury platformy a definice formátů

MPO FV 10202: Databáze materiálových mikrostruktur pro aditivní výrobu

Vypracovali: Š. Beneš, K. Kolářová, M. Doškář, D. Rypl

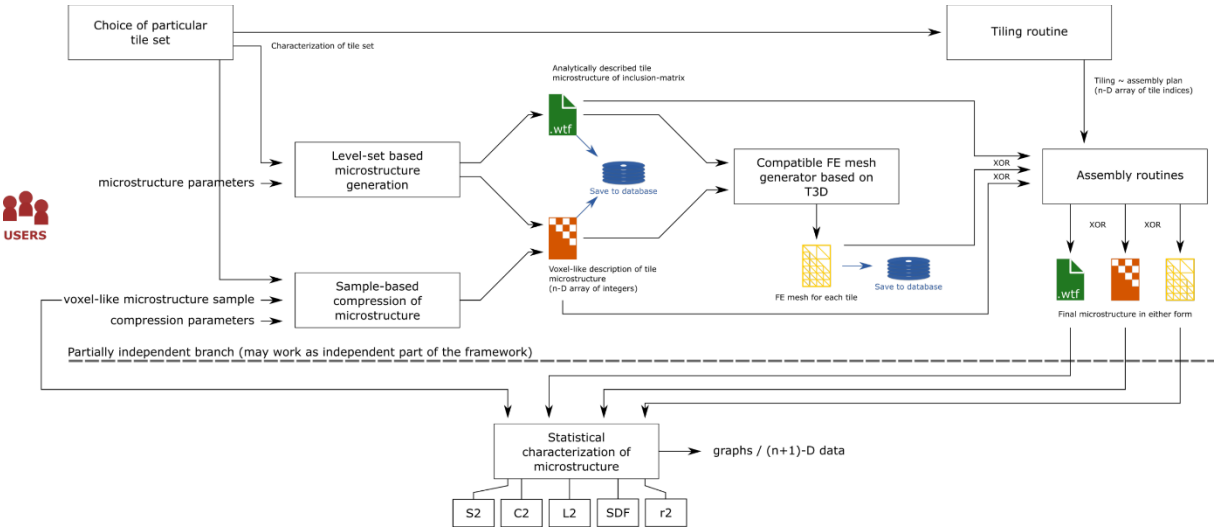
Poslední editace: 12/2020

Obsah

Stručná charakteristika úkolu

Cílem této analýzy bylo navrhnout základní strukturu celé platformy a určit optimální datové formáty na základě identifikovaných požadavků na jednotlivé nástroje. Již při přípravě návrhu projektu bylo rozhodnuto o serverovém řešení platformy, které umožní provozování celé platformy lokálně u zákazníka (on-premises) nebo na veřejně přístupném cloudu (software-as-a-service). Toto řešení bylo po přezkoumání zachováno i nadále, tj. bude vyvinuta serverová část platformy, která bude zajišťovat komunikaci mezi jednotlivými výpočetními nástroji, databází a uživatelským rozhraním. Bylo rozhodnuto, že rozhraní pro přístup a zobrazování dat z databáze i ovládání vyvinutých nástrojů pro charakterizaci, tvorbu a analýzu mikrostrukturálních dat bude dodáno formou webové aplikace pro zajištění co nejširší kompatibility a přenositelnosti a odstranění nutnosti instalace na uživatelském zařízení. Toto webové rozhraní bude komunikovat se serverovou komponentou, která poběží na počítači v interní síti uživatele, případně na vzdáleném počítači ve veřejném cloudu.

Návrh architektury systému



Obrázek 1: Analýza technického řešení platformy GEOSUP

Obrázek 1 znázorňuje výpočetní model sady nástrojů pro charakterizaci a tvorbu mikrostruktur. Obsahuje také výčet vstupů a výstupů z jednotlivých nástrojů. Ze systémové analýzy platformy GEOSUP vyplývá existence celkem pěti jejích základních komponent. Jde o permanentní úložiště popisů mikrostruktur, nástroje pro zpracování mikrostruktur, dočasné úložiště souborů potřebných pro běh těchto nástrojů, aplikační rozhraní zprostředkovávající komunikaci mezi uživatelským rozhraním, nástroji a datovým úložištěm a samotné uživatelské rozhraní.

Technické řešení serverové části

- **SERVEROVÁ ČÁST** poskytující aplikační rozhraní ve formě REST API, se kterým komunikuje webový klient. Implementace bude provedena pomocí webového frameworku ASP.NET Core v jazyce C#.
- **DATABÁZE** popisů materiálových mikrostruktur. Návrh obsahuje adaptér pro standardní relační SQL databázi a rovněž pro No-SQL databázi ukládající přímo JSON dokumenty (např. MongoDB). Rozhodnutí o konkrétním typu databáze bude ovlivněné předpokládanými náklady na provoz.

- **NÁSTROJE** pro zpracování mikrostruktur (generování dlaždic, dláždění, statistiky atd.) realizované jako serverless komponenty (např. Azure Functions), které se spouští až na vyžádání pomocí REST API. Samotné nástroje jsou implementovány v jazyce C či C++ a jsou přeložené jako konzolové aplikace pro OS Windows x64. Popis jednotlivých nástrojů je uveden v následující kapitole.
- **ÚLOŽIŠTĚ VÝSTUPŮ** z nástrojů pro zpracování mikrostruktur. Jde především o dočasné soubory (obrázky, případně JSON dokumenty) potřebné pro běh nástrojů pro zpracování mikrostruktur. Jde o cenově efektivní uložiště s nízkými latencemi pro čtení (read-access geo-redundant blob storage).

Technické řešení klientské části

- **WEBOVÉ ROZHRANÍ** realizované jako SPA (Single Page Application) s využitím webového frameworku Aurelia. Jazyky pro implementaci: HTML, CSS, Typescript.

Hosting

Všechny komponenty systému jsou hostovány ve veřejném cloudu Microsoft Azure. Jednotlivé komponenty jsou mapovány na služby, které poskytuje Azure (Azure web app, Azure Function App, Azure Storage Account, Azure Cosmos DB resp. Azure SQL Database). Pro všechny služby je zvolen region západní Evropa (momentálně nejbližší ČR kvůli nízkým latencím).

Samostatné konzolové aplikace

Vyjma nástrojů pro tvorbu konečněprvkové diskretizace, které využívají existují software T3D a které jsou z důvodu plné kompatibility napsány v jazyce C, jsou ostatní nástroje vyvíjeny v jazyce C++, konkrétně pak v jeho poslední revizi ISO/IEC 14882:2017 (označované též jako C++17). Důvodem tohoto výběru je především ohled na co nejvyšší výpočetní efektivitu. Jednotlivé nástroje jsou napsány jako izolované konzolové aplikace s cílem co největší nezávislosti na cílové platformě a architektuře systému. Tyto aplikace budou spouštěny na vyžádání z aplikačního serveru s využitím serverless výpočetního modelu. Zdrojový kód je psán v angličtině (včetně komentářů) pro zajištění možnosti mezinárodní spolupráce na dalším rozvoji jednotlivých nástrojů. Z důvodu zachování jednotnosti kódů při vývoji, na němž se podílí více autorů, byla pro nástroje psané v jazyce C++ vytvořená doporučení týkající se programovací stylu. Tato doporučení jsou součástí samostatné zprávy.

Níže je uveden soupis jednotlivých nástrojů s popisem jejich hlavní funkcionality a jejich vstupů a výstupů. Detailnější popis bude dodán v manuálech k jednotlivým nástrojům.

1. Nástroj pro charakterizaci mikrostruktury

Cílem nástroje je poskytnout statistickou kvantifikaci rozložení jednotlivých materiálů v mikrostruktuře pomocí standardizovaných postupů běžné využívaných v těchto úlohách. Nástroj bude počítat dvoubodovou pravděpodobnostní funkci a dvoubodovou agregátovou funkci.

Vstupem bude (i) materiálová mikrostruktura zadaná buď formou obrazových dat nebo indexy materiálové fáze na pravidelné mřížce, (ii) zvolené parametry prostorové statistiky a (iii) pomocné parametry nástroje.

Výstupem pak bude spočítána statistika ve zvoleném formátu (předem domluvené formáty výstupu jsou VTK formát ve verzi XML VTI a obecný datový formát platformy založený na formátu JSON).

2. Nástroj pro syntézu komprimovaných dat

Cílem nástroje je poskytnout základní funkcionalitu vyvíjené databáze, tj. umožnit z uložených dat vytvářet materiálové vzorky o uživatelem zadané velikosti. Na základě zadané definice souboru modulů nástroj prvně vytvoří plán jejich rozmístění (tvz. *dláždicí plán*), a to buď s využitím stochastického algoritmu, nebo na základě dodatečné informace poskytnuté ve formě dodatečného obrázku (po namapování poskytnutého obrázku na požadovanou velikost výsledné mikrostrukturální realizace budou pro dané pozice preferovány moduly, které mají podobné objemové zastoupení/světlost jako odpovídající místo v zadaném obrázku). Na základě vytvořeného plánu bude následně syntetizována výsledná mikrostruktura ze zadaných informací o jednotlivých modulech v požadovaném formátu.

Vstupem nástroje tak budou (i) popis souboru modulů, (ii) informace o materiálové mikrostruktuře uložené v modulech, (iii) pomocné parametry nástroje.

Výstupem pak bude (i) dláždicí plán a (ii) mikrostrukturální realizace ve zvoleném formátu.

3. Nástroj pro tvorbu komprimované reprezentace parametrických mikrostruktur

Generování mikrostruktur pomocí Random Sequential Adsorption (RSA) algoritmu je jednou z nejčastějších metod parametrického generování mikrostruktur. Vyvíjená platforma poskytne uživateli tento algoritmus v pokročilé podobě využívající implicitního popisu geometrie pomocí úrovňových množin (tzv. „level sets“). Oproti základnímu algoritmu nedochází ke zpomalování při vyšších objemových zastoupeních, a navíc tento přístup umožňuje generovat komplexní geometrie typově blízké otevřeným a uzavřeným pěnám ze zadaného rozložení částic.

Algoritmus pro tvorbu mikrostruktur bude pracovat ve dvou krocích: v prvním kroku bude umisťovat částice zadaného tvaru, v druhém kroku pak bude na základě rozmístění částic vytvářet komplexní geometrie. Oba kroky budou nezávislé a uživateli bude umožněno využít i jen jeden z nich (buď pro generování částic či se zadáním vlastního rozložení částic).

Vstupem nástroje bude (i) popis souboru modulů, (ii) parametry nástroje.

Výstupem bude buď (i) rozmístění inkluzí v souboru modulů nebo (ii) implicitně definovaná geometrie jednotlivých modulů. Na vyžádání budou ještě generovány výstupy pro následující dva nástroje určené k tvorbě konečněprvkové diskretizace (analyticky popsaná geometrie částic bude vstupem do nástroje č. 5 tohoto seznamu, implicitně definovaná geometrie pak bude vstupovat do nástroje č. 6).

4. Nástroj pro tvorbu komprimované reprezentace z referenčních dat

Tento nástroj je zaměřen na tvorbu komprimované reprezentace materiálové mikrostruktury, tj. na kompresi obsažené mikrostrukturální informace, do formy souboru modulů na základě zadané referenční realizace mikrostruktury. Z této mikrostruktury budou vybrány vhodné vzorky, které budou postupy využívanými v počítačové grafice spojeny s minimalizací viditelných artefaktů do jednotlivých modulů.

Vstupem nástroje budou (i) referenční mikrostruktura, (ii) definice zvoleného souboru modulů a (iii) parametry nástroje.

Výstupem pak bude komprimovaný model mikrostruktury ve formě jednotlivých modulů, pro něž budou data o mikrostruktuře uložena buď jako obrazová informace nebo ve formě materiálových indexů na pravidelné síti.

5. Nástroj pro tvorbu konečněprvkové diskretizace z analytických geometrií

Pro částicové modely mikrostruktury z nástroje č. 3 bude možné vytvořit konečněprvkovou diskretizaci pomocí tohoto nástroje. Hlavní náplní nástroje bude zpracování částicových modelů a nalezení jejich vhodné reprezentace pro následnou diskretizace s ohledem na kvalitu konečněprvkového modelu, eliminaci malých prvků vznikajících na rozhraní jednotlivých modulů a především zajištění kompatibility sítě přes odpovídající si geometrické entity (hrany a stěny) jednotlivých modulů.

Vstupem nástroje bude (i) definice souboru modulů, (ii) rozmístění a analytický popis částic a (iii) parametry diskretizace.

Výstupem pak bude konečněprvkové diskretizace jednotlivých modulů.

6. Nástroj pro tvorbu konečněprvkové diskretizace implicitně zadaných geometrií

Pro konečněprvkovou diskretizaci implicitně definované geometrie jednotlivých modulů bude vyvinut druhý diskretizační nástroj. Před samotnou diskretizací bude hlavní úkolem tohoto nástroje nalézt vhodnou aproximaci rozhraní jednotlivých materiálových fází (podobně jako v případě „marching cubes“ algoritmu), její úpravu vzhledem k výsledné kvalitě sítě a zajištění kompatibility sítě přes odpovídající si geometrické entity jednotlivých modulů. Stejně jako v předchozím případě bude pozornost věnována eliminaci výskytu příliš malých prvků, které by vedly na nárůst výpočetní náročnosti při zpracování takto generovaných diskretizací.

Vstupem nástroje bude (i) definice souboru modulů, (ii) implicitně definovaná geometrie jednotlivých modulů a (iii) parametry diskretizace.

Výstupem bude konečněprvkové diskretizace jednotlivých modulů.

7. Nástroj pro numerickou analýzu

Platforma bude doplněna o nástroje pro numerickou analýzu makroskopické odezvy, tj. výpočet homogenizovaných vlastností, a určení lokálních polí pro jednotlivé moduly i syntetizované mikrostruktury. Tento nástroj bude primárně určen na výpočet lineárních problémů – skalárních transportních procesů i elasticity – pomocí metody konečných prvků a formulace prvního řádu umožňující volbu okrajových podmínek.

Vstupem nástroje bude (i) materiálová mikrostruktura v jednom z definovaných formátů (typ 3, 4 nebo 5 ze seznamu níže), (ii) materiálové parametry každé fáze, (iii) parametry analýzy a (iv) volitelné parametry nástroje.

Výstupem pak budou (i) makroskopické vlastnosti ve formě homogenizované matice vodivosti či tuhosti a (ii) rozložení lokálních termomechanických polí v mikrostruktuře (ve výstupním formátu VTK).

Datové formáty

Pro výměnu dat mezi jednotlivými moduly platformy GEOSUP byla testována řešení využívající některé ze standardizovaných transportních formátů, konkrétně XML, JSON a YAML. Primární datové formáty pro reprezentaci mikrostruktur byly nakonec založeny na JavaScript Object Notation (JSON) formátu, jak je popsáno níže. Jedná se o standardní formát používaný ve webových technologiích. Výhodou je, že stejný dokument lze využít jak pro definici vstupu do nástrojů, tak při přenosu dat mezi REST API a webovým klientem a rovněž pro přímé uložení v indexované No-SQL databázi. Další výhodou je, že se jedná o poměrně úsporný formát (oproti např. XML). Binární výstup z nástrojů je navíc možné zapsat pomocí base64 kódování přímo do JSON dokumentu.

Struktura a požadované záznamy pro jednotlivé datové soubory využívané v rámci vyvíjené platformy jsou uvedeny níže (včetně krátkých ukázek, pokud je potřeba).

1. Definice souboru modulů

Informace o počtu modulů a definice jejich návaznosti/kompatibility je uložena v základním JSON formátu, který obsahuje následující záznamy:

- `setId` – Řetězec znaků odpovídající unikátnímu kódu souboru modulů, kterému dláždicí plán odpovídá (kód je ve formátu UUID)
- `tiles` – Pole JSON objektů uchovávajících informaci o jednotlivých modulech
  - `id` – Celočíselný (nezáporný) unikátní kód modulu (jednotlivé moduly v souboru nemusí být číslovaný spojitě)
  - `codes` – Čtyř- či šestiprvkové pole obsahující kódy přiřazené jednotlivým hranám/stranám určující požadavek návaznosti/kompatibility mikrostruktury

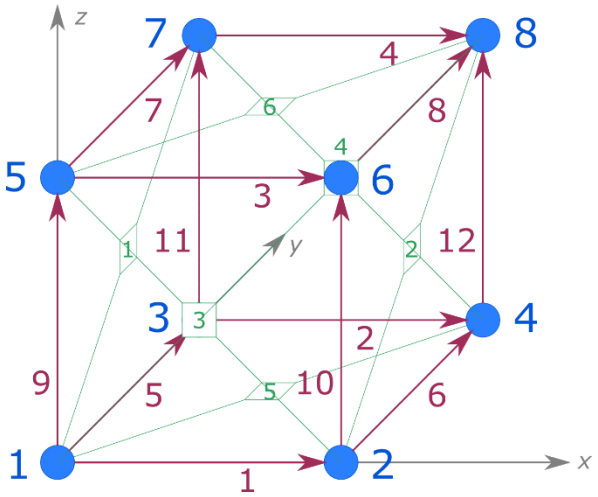
2. Formát pro dláždicí plán

Datový soubor uchovávající informace o dláždicím plánu je postaven na JSON formátu a musí obsahovat následující tři položky:

- `setId` – Řetězec znaků odpovídající unikátnímu kódu souboru modulů (jak je uvedeno v popisu předchozího souboru)
- `size` – Tříprvkové pole udávající velikost dláždění (ve směru x, y a z)
- `data` – Pole celočíselných kódů jednotlivých dlaždic v dláždění. Jednotlivé hodnoty třírozměrného dláždění jsou uvedeny v jednorozměrném poli, kdy první index pozice v dláždění je nejrychleji se měnící. Kód `-1` je vyhrazen pro indikaci prázdné pozice.

Příklad:

```
{
  "setId":          "e641535e-a1db-442e-b01f-63a8bf52ec1c",
  "size":           [ 3, 2, 1 ],
  "data":           [ 1, 0, -1, 1, 1, 0 ]
}
```



Obrázek 1: Číslování geometrických entit hranice modulu

3. Formát pro indexovanou reprezentaci mikrostruktury

Datový soubor obsahuje indexy materiálových fází zadaných na pravidelné mřížce. Soubor je založen na formátu JSON a umožňuje textovou reprezentaci binárních dat kódovaných pomocí base64. Soubor musí obsahovat následující položky:

- `size` – Tříprvkové pole udávající rozlišení mřížky (ve směru x, y a z)
- `compressed` – Logická hodnota (true/false) určující, zda je použito base64 kódování
- `data` – Na základě přechozího pole obsahuje tento záznam
    - Pole celočíselných kódů materiálových fází. Jednotlivé hodnoty třírozměrného dláždění jsou uvedeny v jednorozměrném poli, kdy první index pozice v dláždění je nejrychleji se měnící. Kód -1 je vyhrazen pro indikaci prázdné pozice.
    - Řetězec znaků obsahující výše uvedená data v binární podobě kódované pomocí base64.

4. Formát Verbose JSON pro konečněprvkovou diskretizaci

Proprietární datový soubor obsahující informace o konečněprvkové diskretizaci v uživatelsky přístupném formátu založeném na JSON. Tento formát je určen především pro menší a středně velké úlohy z důvodu jeho větší velikosti (oproti formátu Base64 JSON). Soubor musí obsahovat:

- `nodes` – Pole tříprvkových polí obsahujících souřadnice jednotlivých uzlů diskretizace, pozice v poli slouží jako index jednotlivých uzlů.
- `elements` – Pole JSON objektů odpovídajících jednotlivým prvkům obsahujících
    - `type` – Typ prvku (následuje číslování užívané ve VTK souborech)
    - `matID` – Index materiálové fáze
    - `nodes` – Pole indexů uzlů daného elementu (délka pole závisí na typu elementu, pořadí je dáno pořadím ve VTK formátu)

V případě diskretizace jednotlivých modulů je třeba ještě dodat informaci o pořadí uzlů na jednotlivých geometrických entitách hranice modulu v záznamu `boundary`. U jednotlivých polí záleží na pořadí. Vrcholy, hrany a stěny modulů jsou číslovány, jak je zobrazeno na Obrázek 1. V případě dvoudimenzionálního modulu je uvažována pouze rovina x-y a ostatních entity jsou ponechány prázdné.

- `boundary` – JSON objekt obsahující tři záznamy

- `vertices` – 8-prvkové pole obsahující indexy uzlů ve vrcholech modulu
- `edges` – 12-prvkové pole libovolně dlouhých polí obsahujících indexy uzlů na hranách modulu
- `faces` – 6-prvkové pole libovolně dlouhých polí obsahujících indexy uzlů na stranách modulu

Posledním záznamem `periodicPairs` je možné dodat informaci o dvojících uzlů, které si odpovídají v případě, že jsou geometrie a konečněprvková diskretizace periodické.

- `periodicPairs` – Pole obsahující indexy dvojic uzlů (hlavní uzel a periodický obraz)

Příklad:

```
{
  "nodes": [
        [0.000000, 0.000000, 0.000000],
        [0.500000, 0.000000, 0.000000],
        [1.000000, 0.000000, 0.000000],
        [0.000000, 0.500000, 0.000000],
        [0.400000, 0.600000, 0.000000],
        [1.000000, 0.500000, 0.000000],
        [0.000000, 1.000000, 0.000000],
        [0.500000, 1.000000, 0.000000],
        [1.000000, 1.000000, 0.000000]
  ],
  "elements": [
        { "type": 9, "matID": 0, "nodes": [0, 1, 4, 3] },
        { "type": 9, "matID": 0, "nodes": [1, 2, 5, 4] },
        { "type": 9, "matID": 0, "nodes": [3, 4, 7, 6] },
        { "type": 9, "matID": 0, "nodes": [4, 5, 8, 7] }
  ],
  "boundary": {
        "vertices": [ [0], [2], [6], [8], [], [], [], [] ],
        "edges": [ [3], [5], [], [], [1], [7], [], [], [], [], [], [] ],
        "faces": [ [], [], [], [], [], [] ]
  },
  "periodicPairs": [ [0, 2], [0, 6], [0, 8], [1, 7], [3, 5] ]
}
```

5. Formát Base64 JSON pro konečněprvkovou diskretizaci

Druhý formát používáný v rámci platformy pro ukládání informace o konečněprvkové síti je cílený především na větší úlohy a datovou úsporu. Ideově vychází z formátů VTK, ale je upraven pro potřeby zobrazování a ukládání konečněprvkových výpočtů na serveru. Informace je uložena ve dvou JSON souborech: první soubor s příponou `mesh.json` obsahuje pouze geometrické informace o konečněprvkové diskretizaci; druhý soubor s příponou `attribute.json` obsahuje informace o materiálu přiřazeném každému elementu.

Soubor mesh.json

JSON soubor popisující geometrii konečněprvkové sítě musí obsahovat:

- `Index` – Celočíselný index sítě (ve většině případů 1 jelikož je uvažována pouze jedna síť)
- `NumberOfPoints` – Počet uzlů diskretizace

- `NumberOfCells` – Počet elementů diskretizace
- `Center` – Tříprvkové pole určující (přibližný) geometrický střed diskretizace
- `Radius` – Poloměr kružnice/koule opsané diskretizaci se středem v `Center`
- `PointCoordinates` – Řetězec znaků obsahující data o souřadnicích jednotlivých uzlů ve formátu base64. Vstupní data jsou uložena ve vektoru, kdy každým záznamem je dvou či tříprvkový vektor souřadnic daného uzlu. Jednotlivé hodnoty musí být ve formátu 32-bit single-precision floating point.
- `CellTypes` – Řetězec znaků obsahující data o typu jednotlivých prvků ve formátu base64. Vstupní data jsou uložena ve vektoru, jehož délka odpovídá `NumberOfCells` a jednotlivé komponenty jsou kódy typu prvků (následuje číslování užívané ve VTK souborech) uložené jako 8bitové nezáporné celočíselné hodnoty, tj. char/byte.
- `CellConnectivity` – Řetězec znaků obsahující data o uzlech jednotlivých prvků. Data jsou uložena ve vektoru, jehož každý záznam odpovídá jednomu prvku a obsahuje vektor indexů vrcholů pro zvolený prvek s délkou, která je dána typem prvku. Jednotlivé hodnoty jsou uloženy jako 32bitové celé číslo, tj. Int32.

Soubor attribute.json

JSON soubor popisující vlastnosti, v tomto případě indexy materiálů přiřazených jednotlivým prvkům, geometrických entit v konečněprvkové diskretizaci, musí obsahovat následující položky:

- `MeshIndex` – Celočíselný index odpovídající konečněprvkové sítě, pro níž je soubor vygenerován
- `FieldName` – Název popisovaných vlastností
- `Location` – Umístění dat („Cells“ nebo „Points“)
- `Encoding` – JSON objekt popisující uložená data
    - `DataType` – Název datového typu („Float32“, „Float64“, „Int32“, nebo „UInt8“)
    - `OriginalLength` – Původní délka pole
    - `Offset` – Index první pozice v původním poli, kde se nevyskytuje `DefaultValue`
    - `Length` – Skutečná délka uložených dat (po vynechání opakujících se počátečních a koncových `DefaultValue` hodnot)
    - `DefaultValue` – Hodnota, kterou je vyplněn vektor v případě vynechání opakujících se počátečních a koncových hodnot
- `Data` – Řetězec znaků obsahující data v base64 kódování o délce `Length`

Příklad mesh.json:

```
{
  "Index":              1,
  "NumberOfPoints":     434,
  "NumberOfCells":      324,
  "Center":             [0.6375, 0.095, 0.16],
  "Radius":             0.6719607,
  "PointCoordinates":    "//9/MwAAADIK16M+//9/MwAAADJvEoM+gDS...",
  "CellConnectivity":   "SwAAADgAAAA0AAAASQAAAD4AAAArAAAAKAA...",
  "CellTypes":           "DAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDA..."
}
```

Příklad attribute.json:

```
{
  "MeshIndex":      1,
  "FieldName":      "material",
  "Location":       "Cells",
  "Compression":    null,
  "Encoding": {
   "DataType":      "Int32",
   "OriginalLength": 324,
   "Offset":        160,
   "Length":        164,
   "DefaultValue":  "18"
  },
  "Data":           "EwAAABMAAAATAAAAEwAAABMAAAATAAAAEwAAABMAAAATAA..."
}
```

6. Formát popisující rozmístění částic v souboru modulů

Rozmístění částic je uloženo pro další zpracování v datovém souboru založeném na JSON formátu, který přebírá základní strukturu z formátu popisujícího soubor modulů (může tedy být využit i jako tento soubor) a k definici každého modulu přidává záznam o částicích v tomto modulu umístěných. Soubor obsahuje následující záznamy:

- `setId` – Řetězec znaků odpovídající unikátnímu kódu souboru modulů
- `tiles` – Pole JSON objektů uchovávajících informaci o jednotlivých modulech
    - `id` – Celočíselný (nezáporný) unikátní kód modulu
    - `codes` – Čtyř či šestiprvkové pole obsahující kódy přiřazené jednotlivým hranám/stranám určující požadavek návaznosti/kompatibility mikrostruktury
    - `particles` – Pole JSON objektů obsahující záznamy pro každou částici obsaženou v modulu. Tento záznam obsahuje tři části: povinné prvky, podmíněně povinné prvky a volitelné/pomocné prvky

Povinné záznamy pro JSON objekt částice

- `centre` – Tříprvkové pole obsahující x, y a z souřadnice středu částice
- `circumscribedRadius` – Poloměr kružnice/koule opsané dané částici
- `type` – Celočíselný kód typu částice (kruh = 1, elipsa = 2, mnohoúhelník = 3, koule = 4, elipsoid = 5, mnohostěn = 6)

Na základně typu částice jsou vyžadovány následující prvky JSON objektu:

- kruh:
    - `radius` – Poloměr částice
- elipsa
    - `semiAxes` – Dvouprvkové pole obsahující velikost hlavní a vedlejší poloosy
    - `basis` – Pole devíti hodnot obsahující bázové vektory (zapsané sekvenčně) určující orientaci elipsy v prostoru (konkrétně v rovině x-y prostoru)
- mnohoúhelník
    - `vertices` – Pole obsahující souřadnice vrcholů mnohoúhelníku zadané relativně ke středu částice. Vrcholy jsou řazené proti směru hodinových ručiček.
- koule

    - `radius` – Poloměr částice
- elipsoid
    - `semiAxes` – Tříprvkové pole obsahující velikosti poloos (od největší po nejmenší)
    - `basis` – Pole devíti hodnot obsahující bázové vektory (zapsané sekvenčně) určující orientaci elipsoidu v prostoru. První je uveden směr hlavní poloosy, poslední je uveden směr nejmenší poloosy.
- mnohostěn
    - `vertices` – Pole obsahující souřadnice vrcholů mnohostěnu zadané relativně ke středu částice
    - `faces` – Pole obsahující popis jednotlivých stěn pomocí trojúhelníků; pole obsahuje trojice indexů vrcholů definujích jeden trojúhelník s normálou směřující ven z mnohostěnu

Volitelné záznamy pro JSON objekt částice

Tyto záznamy není nutné ručně zadávat při tvorbě soupisu částic mimo nástroj. Záznamy jsou vypsány pouze v případě, že byl použit tento nástroj i pro generování soupisu částic, a slouží ke kontrole a detailnějšímu přehledu o vygenerovaných datech.

- `isMaster` – Logická hodnota určující, zda byla tato částice umístěna jako první
- `isVirtual` – Logická hodnota určující, zda se jedná o virtuální částici (tj. částici mimo oblast)
- `uniqueID` – Celočíselný kód částice (unikátní pro každou částici)
- `masterID` – Celočíselný kód sdílený všemi periodickými kopiemi stejné částice
- `intersections` – Pole o 4/6 logických hodnotách určující, zda daná částice protíná hrany/stěny modulu ve 2D/3D případě
- `originType` – Celočíselný kód určující typ entity, která zapříčinila periodické rozkopírování (modul = 1, stěna = 2, hrana = 3, vrchol = 4)
- `originIndex` – Celočíselný kód entity, která zapříčinila periodické rozkopírování částice (pokud `originType` = 1, jedná se o `id` modulu)