Deriving the Approximate Algorithm

Operator splitting and approximation.

- A method to approximate e^{tW} is a first-order operator splitting algorithm that imposes a domain decomposition by means of an expanded cell complex.
- The decomposition corresponds to summing operators, $W = \sum_{(d)} W_{(d)} = \sum_{(d,c)} W_{(c,d)}$ over pre-expansion dimensions *d*, and cells *c* of each dimension where $d\downarrow$ means we multiply from right to left in order of highest dimension to lowest:

$$e^{tW} \approx \left(\prod_{d\downarrow} e^{\frac{t}{n} W_{(d)}} \right)^{n \to \infty}$$

$$e^{t'W_{(d)}} = \prod_{c \subset d} e^{t'W_{(c,d)}} \quad \text{where} \quad [W_{(c,d)}, W_{(c',d)}] \approx 0 \quad \text{and} \quad t' \equiv \frac{t}{n}$$

$$W_{(c,d)} = \sum_r W_{r,c} \equiv \sum_r \sum_{\left\{ \begin{array}{c} R \mid \varphi(R) = c, \\ R \text{ instantiates } r \end{array} \right\}} W_r(\, R \mid c, d)$$

Yet Another Graph Library (YAGL)

Brief Overview

- Simple header-only C++17 library.
- Nodes make use of special variant nodes to to take on different types of data.
  - A further specialization is defined for DGGML.
- Nodes are stored in a map for quick lookup.
- Includes a few algorithms such as finding connected components and finding graph isomorphisms.

```
template <typename ... Types>
struct VariantData
{
    using node_variant_t = std::variant<Types ...>;
    node_variant_t data;
}
```

Base type for the variant data in YAGL.

```
template <typename ... Types>
struct SpatialData3D : VariantData<Types ...>
{
    double position[3];
}
```

Specialization of the variant data type for DGGML.

```
struct Negative
{
    double velocity[3];
    double unit_vec[3];
};
```

Example of a type.

```
struct Positive
{
    double velocity[3];
    double unit_vec[3];
};
```

Example of another type.

```
using graph_type =
    YAGL::Graph<unsigned int,
    SpatialNode3D<Negative, Positive>>;
```

Graph defining interface.