1 Installation and Usage

1. Download the latest version of GAPoTNumLib from GitHub: https://github.com/ga-explorer/GAPoTNumLib
2. Open and Build the GAPoTNumLib solution using x64 Debug configuration. Make sure the GAPoTNumLib.Framework project is the default project of the solution
3. In the MATLAB toolbox open the file gapotInit.m and edit the variable gapotAssemblyPath to be the path containing the GAPoTNumLib.Framework.exe file
4. In MATLAB add the toolbox folder to the MATLAB path
5. You can find examples for using the toolbox in files gapotSample1.m, gapotSample2.m, etc.

2 Representation of GAPoT Multivectors

The design of GAPoTNumLib mainly targets the representation and manipulation of sparse Euclidean multivectors containing elements of grades 0,1, and 2. Contrast to most general purpose GA libraries, the dimension of vectors in GAPoTNumLib is arbitrary, and can be in the range of thouthands. Additionally, creation of GAPoTNumLib vectors is formulated to be as close as possible to GAPoT symbolic representation of current and voltage vectors. Other GAPoT multivectors are constructed using the geometric product of GAPoT vectors.

A current or voltage vector in GAPoT is essentially a sum of n polar phasors of the form:

$$V = \sum_{i=1}^{n} \alpha_i \mathrm{exp}\left(\theta_i \sigma_{2i-1}\sigma_{2i}\right)\sigma_{2i-1} \tag{1}$$

Where α_i, θ_i are magnitudes and angles of individual phasors. We can also re-write this into two equivalent forms. The rectangular phasor form is:

$$\begin{aligned} V &= \sum_{i=1}^{n} \alpha_i \left(\mathrm{cos}\theta_i + \mathrm{sin}\theta_i \sigma_{2i-1}\sigma_{2i}\right)\sigma_{2i-1} \\ &= \sum_{i=1}^{n} \left(x_i + y_i \sigma_{2i-1}\sigma_{2i}\right)\sigma_{2i-1} \end{aligned} \tag{2}$$

Where $x_i = \alpha_i \mathrm{cos}\theta_i, y_i = \alpha_i \mathrm{sin}\theta_i$ are cartesian components of the phasor. The third form is the most similar to traditional representation of multivectors in GA libraries as a sum of scaled basis blades:

$$\begin{aligned} V &= \sum_{i=1}^{n} \left(x_i \sigma_{2i-1} + y_i \sigma_{2i-1}\sigma_{2i}\sigma_{2i-1}\right) \\ &= \sum_{i=1}^{n} \left(x_i \sigma_{2i-1} - y_i \sigma_{2i}\right) \\ &= \sum_{i=1}^{2n} v_i \sigma_i \end{aligned} \tag{3}$$

Where $v_i = x_i = \alpha_i \cos\theta_i$ for $i = 1, 3, 5, \ldots, 2n-1$, and $v_i = -y_i = -\alpha_i \sin\theta_i$ for $i = 2, 4, 6, \ldots, 2n$.

In GAPoTNumLib, a simple textual representation can be used to construct GAPoT vectors using either 3 forms: sum of polar phasors, rectangular phasors, or terms. Internally, however, all GAPoT vectors are stored as a sparse list of terms and other forms are composed and displayed to the user on demand. The following are examples for the textual representation of vectors in GAPoTNumLib:

- `'-1.3<1>, 1.2<3>, -4.6<6>'` represents a GAPoT vector in sum of terms form: $-1.3\sigma_1 + 1.2\sigma_3 - 4.6\sigma_6$
- `'p(233.92, -90) <1,2>, p(-120, 30) <3,4>'` represents a GAPoT vector in sum of polar phasors form: $233.92e^{-90^{\circ}\sigma_1\sigma_2}\sigma_1 - 120e^{30^{\circ}\sigma_3\sigma_4}\sigma_3$
- `'r(10, 20) <1,2>, r(30, 0) <3,4>'` represents a GAPoT vector in sum of rectangular phasors form: $(10 + 20\sigma_1\sigma_2)\,\sigma_1 + (30 + 0\sigma_3\sigma_4)\,\sigma_3 = (10\sigma_1 - 20\sigma_2) + (30\sigma_3)$ which is equivalent to `'10<1>, -20<2>, 30<3>'`
- `'1<1>, r(10, 20) <2,3>, p(-120, 30) <4,5>'` represents a GAPoT vector in sum of phasors form: $1\sigma_1 + (10 + 20\sigma_3\sigma_4)\,\sigma_3 - 120e^{30^{\circ}\sigma_5\sigma_6}\sigma_5$

The geometric product of two GAPoT vectors is a multivector containing only grade 0 and 2 elements. In GAPoTNumLib such multivectors are called biversors, as they are the geometric product of two vectors. The user can also create a biversor from a textual representation such as `'3<>, -2<1,2>, 4<3,4>'` which represents the multivector $3 - 2\sigma_1\sigma_2 + 4\sigma_3\sigma_4$.

3 GAPoTNumLib Classes

In the .NET solution, the user can find several classes to represent GAPoT multivectors under the `GAPoTNumLib.GAPoT` namespace as follows:

- `GaPoTNumVector` is the class used to represent GAPoT vectors, typically holding currents and voltages. The main operations this class provides include setting, getting, and adding terms, polar phasors, and rectangular phasors. The user can also add and subtract two GAPoT vectors, compute their geometric product, find the negative, norm, squared norm, reverse, and inverse of a GAPoT vector. In addition, several methods for displaying the GAPoT vector in various text and LaTeX formats exist.

- `GaPoTNumBiversor` is used to represent GAPoT biversors; sparse multivectors which only contains elements of grades 0 and 2 typically representing power and impedance. The class provides methods for setting, getting, and adding individual terms of grade 0 and 2. The user can also compute the negative, norm, squared norm, reverse, and inverse of biversors. There are several methods for extracting power quantities from the biversor such as asctive, non-active, reactive, fundamental reactive, and harmful power parts. In addition, several methods for displaying the GAPoT biversor in various text and LaTeX formats exist.
- `GaPoTNumMultivector` is capable of representing arbitrary sparse multivectors containing terms of grades < 32. A vector or biversor can be converted into a multivector using their `ToMultivectr()` methods. The user can extract the vector part (grade 1 terms) of a multivector using the `GetVectorPart()` method, and the same for the `GetBiversorPart()` method. Many computations on multivectors are also implemented like the geometric, outer, left-contraction, and scalar products using the methods `Gp()`, `Op()`, `Lcp()`, and `Sp()` in addition to the reverse, inverse, negative, squared norm, etc.

For the elements of a GAPoT vector, there are three classes that can be used to hold information on terms and phasors:

- `GaPoTNumVectorTerm` holds a single term vector.
- `GaPoTNumPolarPhasor` holds a single polar phasor vector.
- `GaPoTNumRectPhasor` holds a single rectangular phasor vector.

4 Operations on Multivectors

4.1 Construction and Update Operations

4.2 Mathematical Operations

4.3 Text Operations

4.4 MATLAB Interoperability