|  |  |
| --- | --- |
|  | **2011** |
|  | University of Colorado at Denver, Colorado School of Public Health  Sarah Kreidler, Jonathan Cohen |

|  |
| --- |
| **GLIMMPSE: Matrix Service REST API** |
| The Glimmpse software system provides a web based user interface to estimate power and sample size for the general linear multivariate model (GLMM) with or without a baseline covariate. The Matrix Service is a RESTful Java web service which can perform matrix operations. The web service will be used primary by other programs, such as the GLIMMPSE web interface, to perform matrix operations not easily performed in a client side browser environment. |

Contents

[**GLIMMPSE: Matrix Service REST API** 1](#_Toc289269358)

[Introduction 3](#_Toc289269359)

[Service End Points: 3](#_Toc289269360)

[Matrix legend 3](#_Toc289269361)

[1. Matrix Addition 3](#_Toc289269362)

[2. Matrix Subtraction 4](#_Toc289269363)

[3. Matrix Multiplication 4](#_Toc289269364)

[3.1. Scalar multiplication 5](#_Toc289269365)

[3.2. Element-wise 5](#_Toc289269366)

[3.3. Matrix multiplication 6](#_Toc289269367)

[3.4. Horizontal direct product 6](#_Toc289269368)

[3.5. Kronecker Products 7](#_Toc289269369)

[4. Matrix Inversion 7](#_Toc289269370)

[5. Rank 7](#_Toc289269371)

[6. Trace 8](#_Toc289269372)

[7. Positive Definite 8](#_Toc289269373)

[8. Orthogonal Polynomial Contrasts 8](#_Toc289269374)

[8.1. Singly repeated measures 8](#_Toc289269375)

[8.2. Double repeated measures 9](#_Toc289269376)

[8.3. Triply repeated measures 9](#_Toc289269377)

[9. Cholesky Decomposition 9](#_Toc289269378)

[10. Vec 10](#_Toc289269379)

[11. Vech 10](#_Toc289269380)

[12. Error Handling 10](#_Toc289269381)

# Introduction

The Glimmpse software system provides a web based user interface to estimate power and sample size for the general linear multivariate model (GLMM) with or without a baseline covariate. The Glimmpse system consists of the following main components:

* Glimmpse.com user interface - a JavaScript/GWT front-end user interface
* Power Service - RESTful Java web service which processes power/sample size requests
* Chart Service - RESTful Java web service which produces graphs. For the Glimmpse system, this service produces power curves.
* File Service - RESTful Java web service providing upload/save functionality for study design information
* **Matrix Service** – RESTful Java web service which can perform matrix operations. The web service will be used primary by other programs, such as the GLIMMPSE web interface, to perform matrix operations not easily performed in a client side browser environment.
* JavaStatistics library - low level library containing routines for computing GLMM power

This document describes the REST API for the Matrix Service. It assumes basic familiarity with [HTTP](http://www.w3.org/Protocols/), [REST](http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm), and the GLMM. For additional background on GLMM power calculations when controlling for a baseline covariate, please see [Glueck & Muller 2003](http://www.ncbi.nlm.nih.gov/pubmed/12898543)

A matrix is a rectangular array of numbers. Matrices will be named with capital letters (A), and elements will be named as lower case letters with row and column subscripts (aij). Dimensions will be expressed as r x c to indicate an array with r rows and c columns. A full description of matrix terminology and operations is found in Chapter 1 of Muller & Stewart.

# Service End Points:

|  |  |  |
| --- | --- | --- |
| **Variable** | **Type** | **Description** |
| name | String | The name of the matrix. |
| rows | Integer | Number of rows in the matrix |
| columns | Integer | Number of columns in the matrix |
| Ar,c | Real | Value of cell r,c in the matrix, where ‘r’ is the row number, and ‘c’ is the column number. |

## Matrix legend

## Matrix Addition

Matrix addition takes two *r x c* matrices, A and B as input and return the element-wise sum of the matrices. Both matrices must have the same number of rows and columns.

* 1. URI: POST /matrix/addition
  2. XML Request:

<matrixList>

<matrix name='*A*' rows='*r*' columns='c'>

<r><c>*a1,1*</c><c>*a1,2*</c>...<c>*a1,c*</c></r>

...

<r><c>*ar,1*</c><c>*ar,2*</c>...<c>*ar,c*</c></r>

</matrix>

<matrix name='*B*' rows='*r*' columns='*c*'>

<r><c>*b1,1*</c><c>*b1,2*</c>...<c>*b1,c*</c></r>

...

<r><c>*br,1*</c><c>*br,2*</c>...<c>*br,c*</c></r>

</matrix>

</matrixList>

* 1. XML Response:

<matrix name=’sum’ rows='*r*' columns='*c*'>

<r><c>*s1,1*</c><c>*s1,2*</c>...<c>*s1,c*</c></r>

...

<r><c>*sr,1*</c><c>s*r,2*</c>...<c>*sr,c*</c></r>

</matrix>

## Matrix Subtraction

Matrix subtraction takes two *r x c* matrices, A and B as input and return the element-wise difference of the matrices.

* 1. URI: POST /matrix/subtraction
  2. XML Request:

<matrixList>

<matrix name='*A*' rows='*r*' columns='*c*'>

<r><c>*a1,1*</c><c>*a1,2*</c>...<c>*a1,c*</c></r>

...

<r><c>*ar,1*</c><c>*ar,2*</c>...<c>*ar,c*</c></r>

</matrix>

<matrix name='*B*' rows='*r*' columns='*c*'>

<r><c>*b1,1*</c><c>*b1,2*</c>...<c>*b1,c*</c></r>

...

<r><c>*br,1*</c><c>*br,2*</c>...<c>*br,c*</c></r>

</matrix>

</matrixList>

* 1. XML Response:

<matrix name='*difference*' rows='*r*' columns='*c*'>

<r><c>*d1,1*</c><c>*d1,2*</c>...<c>*d1,c*</c></r>

...

<r><c>*dr,1*</c><c>*dr,2*</c>...<c>*dr,c*</c></r>

</matrix>

## Matrix Multiplication

The Matrix Web Service supports the 5 types of matrix multiplication as defined in Muller & Stewart (2006). These include:

### Scalar multiplication

Scalar multiplication takes a single *r x c* matrix, A, and a scalar multiplier, b. It returns a single matrix: *Ab = {baij}*.

* + 1. URI: POST /matrix/mult/scalar
    2. XML Request:

<scalarMultiplicationParams>

<matrix name='A' rows='*r*' columns='*c*'>

<r><c>*a1,1*</c><c>*a1,2*</c>...<c>*a1,c*</c></r>

...

<r><c>*ar,1*</c><c>*ar,2*</c>...<c>*ar,c*</c></r>

</matrix>

<scalarMultiplier value=”b”/>**NOTE quotation marks**

</scalarMultiplicationParams>

* + 1. XML Response:

<matrix name='product' rows='*r*' columns='*c*'>

<r><c>*p1,1*</c><c>*p1,2*</c>...<c>*p1,c*</c></r>

...

<r><c>*pr,1*</c><c>*pr,2*</c>...<c>*pr,c*</c></r>

</matrix>

### Element-wise

Element-wise multiplication takes two *r x c* matrices A and B, and returns a single matrix: *A#B = {aijbij}.* The matrices A and B have equal dimensions.

* + 1. URI: POST /matrix/mult/elementWise
    2. XML Request:

<matrixList>

<matrix name='*A*' rows='*r*' columns='*c*'>

<r><c>*a1,1*</c><c>*a1,2*</c>...<c>*a1,c*</c></r>

...

<r><c>*ar,1*</c><c>*ar,2*</c>...<c>*ar,c*</c></r>

</matrix>

<matrix name='*B*' rows='*r*' columns='*c*'>

<r><c>*b1,1*</c><c>*b1,2*</c>...<c>*b1,c*</c></r>

...

<r><c>*br,1*</c><c>*br,2*</c>...<c>*br,c*</c></r>

</matrix>

</matrixList>

* + 1. XML Response:

<matrix name='product' rows='*r*' columns='*c*'>

<r><c>*p1,1*</c><c>*p1,2*</c>...<c>*p1,c*</c></r>

...

<r><c>*pr,1*</c><c>*pr,2*</c>...<c>*pr,c*</c></r>

</matrix>

### Matrix multiplication

Matrix multiplication takes two matrices A (*r x c*) and B (*c x d*) and returns a single matrix: *AB = {cjk},* where

The columns of A must match the rows of B.

* + 1. URI: POST /matrix/mult/
    2. XML Request:

<matrixList>

<matrix name='*A*' rows='*r*' columns='*c*'>

<r><c>*a1,1*</c><c>*a1,2*</c>...<c>*a1,c*</c></r>

...

<r><c>*ar,1*</c><c>*ar,2*</c>...<c>*ar,c*</c></r>

</matrix>

<matrix name='*B*' rows='*c*' columns='*d*'>

<r><c>*b1,1*</c><c>*b1,2*</c>...<c>*b1,c*</c></r>

...

<r><c>*br,1*</c><c>*br,2*</c>...<c>*br,c*</c></r>

</matrix>

</matrixList>

* + 1. XML Response:

<matrix name='product' rows='*r*' columns='*c*'>

<r><c>*a1,1*</c><c>*a1,2*</c>...<c>*a1,c*</c></r>

...

<r><c>*ar,1*</c><c>*ar,2*</c>...<c>*ar,c*</c></r>

</matrix>

### Horizontal direct product

The horizontal direct product takes two matrices A (*r x c*) and B (*r x d*) and returns a single *r x cd* matrix representing element-wise multiplication of pairs of columns of A and B. The matrices A and B must have the same number of rows.

* + 1. URI: POST /matrix/mult/horizontalDir
    2. XML Request:

<matrixList>

<matrix name='*A*' rows='*r*' columns='*c*'>

<r><c>*a1,1*</c><c>*a1,2*</c>...<c>*a1,c*</c></r>

...

<r><c>*ar,1*</c><c>*ar,2*</c>...<c>*ar,c*</c></r>

</matrix>

<matrix name='*B*' rows='*r*' columns='*d*'>

<r><c>*b1,1*</c><c>*b1,2*</c>...<c>*b1,c*</c></r>

...

<r><c>*br,1*</c><c>*br,2*</c>...<c>*br,c*</c></r>

</matrix>

</matrixList>

* + 1. XML Response:

<matrix name='product' rows='*r*' columns='*c*'>

<r><c>*p1,1*</c><c>*p1,2*</c>...<c>*p1,c*</c></r>

...

<r><c>*pr,1*</c><c>*pr,2*</c>...<c>*pr,c*</c></r>

</matrix>

### Kronecker Products

The Kronecker product takes two matrices A (*r x c*) and B (s *x d*) and returns a single *rs x cd* matrix representing the Kronecker or direct product of the two matrices.

* + 1. URI: POST /matrix/mult/kronecker
    2. XML Request:

<matrixList>

<matrix name='*A*' rows='*r*' columns='*c*'>

<r><c>*a1,1*</c><c>*a1,2*</c>...<c>*a1,c*</c></r>

...

<r><c>*ar,1*</c><c>*ar,2*</c>...<c>*ar,c*</c></r>

</matrix>

<matrix name='*B*' rows='*s*' columns='*d*'>

<r><c>*b1,1*</c><c>*b1,2*</c>...<c>*b1,c*</c></r>

...

<r><d>*br,1*</c><c>*br,2*</c>...<c>*br,c*</c></r>

</matrix>

</matrixList>

* + 1. XML Response:

<matrix name='product' rows='*rs*' columns='*cd*'>

<r><c>*p1,1*</c><c>*p1,2*</c>...<c>*p1,cd*</c></r>

...

<r><c>*prs,1*</c><c>*prs,2*</c>...<c>*prs,cd*</c></r>

</matrix>

## Matrix Inversion

Matrix inversion is available for square matrices. This operation takes a single *p x p* matrix and returns its inverse. If the matrix is singular, an error message will be returned.

* + 1. URI: POST /matrix/inverse
    2. XML Request:

<matrix name='A' rows='*p*' columns='*p*'>

<r><c>*a1,1*</c><c>*a1,2*</c>...<c>*a1,p*</c></r>

...

<r><c>*ap,1*</c><c>*ar,2*</c>...<c>*ap,p*</c></r>

</matrix>

* + 1. XML Response:

<matrix name='inverse' rows='*p*' columns='*c*'>

<r><c>*i1,1*</c><c>*i1,2*</c>...<c>*i1,p*</c></r>

...

<r><c>*ip,1*</c><c>*ir,2*</c>...<c>*ip,p*</c></r>

</matrix>

## Rank

The rank operation takes a single *r x c* matrix and returns the number of linearly independent columns of the matrix.

* + 1. URI: POST /matrix/rank
    2. XML Request:

<matrix name='A' rows='*r*' columns='*c*'>

<r><c>*a1,1*</c><c>*a1,2*</c>...<c>*a1,c*</c></r>

...

<r><c>*ar,1*</c><c>*ar,2*</c>...<c>*ar,c*</c></r>

</matrix>

* + 1. XML Response:

<rank value=’n’/>

## Trace

The trace takes a single square matrix (*p x p*) and returns its trace tr(A) =

* + 1. URI: POST /matrix/trace
    2. XML Request:

<matrix name='A' rows='*p*' columns='*p*'>

<r><c>*a1,1*</c><c>*a1,2*</c>...<c>*a1,p*</c></r>

...

<r><c>*ap,1*</c><c>*ar,2*</c>...<c>*ap,p*</c></r>

</matrix>

* + 1. XML Response:

<trace value=’n’/>

## Positive Definite

The positive definite operation takes a single square matrix (*p x p*) and returns a true or false value indicating if the matrix is positive definite.

* + 1. URI: POST /matrix/positiveDefinite
    2. XML Request:

<matrix name='A' rows='*p*' columns='*p*'>

<r><c>*a1,1*</c><c>*a1,2*</c>...<c>*a1,p*</c></r>

...

<r><c>*ap,1*</c><c>*ar,2*</c>...<c>*ap,p*</c></r>

</matrix>

* + 1. XML Response:

<positiveDefinite value=’true|false’/>

## Orthogonal Polynomial Contrasts

The matrix web service produces orthogonal polynomial contrasts for both between and within-subject factors. This operation takes a list of factors as input. For each factor, the user must specify a name, *fi*, and a list of values indicating the spacing of measurements across the factor. For example, if repeated measurements were taken at 1, 4, and 10 weeks, the user would specify a factor name of “weeks” and a value list of 1, 4, and 10.

The matrix service will return a list of contrast matrices, including the grand mean contrast (all zero order trends), all main effect contrasts, and all interaction contrasts. Within-subject contrasts are returned by default. If between-subject contrasts are requested, then the transpose of the equivalent within-subject contrasts will be returned.

* + 1. URI: POST /matrix/contrast/orthopoly
    2. XML Request:

The user will request an orthogonal polynomial contrast by specifying the list of within or between-subject factors for the contrast. Each factor must include a name, and list of values which indicate the spacing of measurements for the factor.

<factorList type=’(between|within)’>

<factor name=’*f1*’><v>*v1*</v>…<v>*vn1*</v></factor>

<factor name=’*f2*’><v>*v1*</v>…<v>*vn2*</v></factor>

…

<factor name=’*fm*’><v>*v1*</v>…<v>*vnm*</v></factor>

</factorList>

* + 1. XML Response:

The matrix service respond with a list of orthogonal polynomial contrasts. Three classes of contrasts will be included in the response:

* Grand mean – contrast for testing hypotheses about the grand mean
* Main effect – contrast for testing main effects hypotheses about a given factor
* Interaction – contrast for testing interaction hypotheses about two or more factors

A contrast definition consists of two main pieces: the contrast matrix, and the list of factors included in the contrast. The list of factors is used by the calling application to determine which factors are included in a main effect or interaction contrast. The grand mean contrast does not include a list of factors.

<orthogonalPolynomialContrastList>

<contrast type=’grandMean’>

<matrix name='grandMean' rows='*r*' columns='*c*'>

<r><c>*v1,1*</c><c>*v1,2*</c>...<c>*v1,c*</c></r>

...

<r><c>*vr,1*</c><c>*vr,2*</c>...<c>*vr,c*</c></r>

</matrix>

</contrast>

<contrast type=’mainEffect’>

<matrix name='*f1*' rows='*r*' columns='*c*'>

<r><c>*v1,1*</c><c>*v1,2*</c>...<c>*v1,c*</c></r>

...

<r><c>*vr,1*</c><c>*vr,2*</c>...<c>*vr,c*</c></r>

</matrix>

<factorList>

<factor name=’*f1*’><v>*v1*</v>…<v>*vn1*</v></factor>

</factorList>

</contrast>

…

<contrast type=’interaction’>

<matrix name='*f1f2…fn*' rows='*r*' columns='*c*'>

<r><c>*v1,1*</c><c>*v1,2*</c>...<c>*v1,c*</c></r>

...

<r><c>*vr,1*</c><c>*vr,2*</c>...<c>*vr,c*</c></r>

</matrix>

<factorList>

<factor name=’*f2*’><v>*v1*</v>…<v>*vn2*</v></factor>

…

<factor name=’*fm*’><v>*v1*</v>…<v>*vnm*</v></factor>

</factorList>

</contrast>

</orthogonalPolynomialContrastList>

## Cholesky Decomposition

This operation takes a single square matrix that is both symmetric and positive definite, and returns two matrices: the matrix representing its square root (L), and its transpose (L’).

* + 1. URI: POST /matrix/decomposition/cholesky
    2. XML Request Entity Body:

<matrix name='A' rows='*p*' columns='*p*'>

<r><c>*a1,1*</c><c>*a1,2*</c>...<c>*a1,p*</c></r>

...

<r><c>*ap,1*</c><c>*ar,2*</c>...<c>*ap,p*</c></r>

</matrix>

* + 1. XML Response:

<choleskyDecomposition>

<matrix name='*L*' rows='*r*' columns='*c*'>

<r><c>*l1,1*</c><c>*l1,2*</c>...<c>*l1,c*</c></r>

...

<r><c>*lr,1*</c><c>*lr,2*</c>...<c>*lr,c*</c></r>

</matrix>

<matrix name='*lTranspose*' rows='*r*' columns='*c*'>

<r><c>*lt1,1*</c><c>*lt1,2*</c>...<c>*lt1,c*</c></r>

...

<r><c>*ltr,1*</c><c>*ltr,2*</c>...<c>*ltr,c*</c></r>

</matrix>

</choleskyDecomposition>

## Vec

The Vec operation takes an *r x c* matrix, A, and returns an *rc x 1* matrix in which the columns of A have been vertically concatenated (see pages 4-5, Muller & Stewart).

* + 1. URI: POST /matrix/vec
    2. XML Request Entity Body:

<matrix name='A' rows='*r*' columns='*c*'>

<r><c>*a1,1*</c><c>*a1,2*</c>...<c>*a1,c*</c></r>

...

<r><c>*ar,1*</c><c>*ar,2*</c>...<c>*ar,c*</c></r>

</matrix>

* + 1. XML Response:

<matrix name='vec' rows='*r*' columns='*1*'>

<r><c>*v1,1*</c></r>

...

<r><c>*vrc,1*</c></r>

</matrix>

## Vech

The Vech operation takes an *r x c* matrix, A, and returns a *n x1* vector containing the unique elements of A (see pages 4-5, Muller & Stewart).

* + 1. URI: POST /matrix/vech
    2. XML Request Entity Body:

<matrix name='A' rows='*r*' columns='*c*'>

<r><c>*a1,1*</c><c>*a1,2*</c>...<c>*a1,c*</c></r>

...

<r><c>*ar,1*</c><c>*ar,2*</c>...<c>*ar,c*</c></r>

</matrix>

* + 1. XML Response:

<matrix name='vech' rows='*r*' columns='*1*'>

<r><c>*v1,1*</c></r>

...

<r><c>*vn,1*</c></r>

</matrix>

## A Note on Error Handling

Any time an http success code (200 or 2xx )is not returned, the code will include a human readable error message along with the appropriate http error code, including:

* The requested matrix operation is not supported
* The input matrices are incorrectly specified
* The dimensions of the matrices are inappropriate for the desired operation
* The operation cannot be performed for the specified matrix (for example, the user requests positive definite check on a non-square matrix)
* General server errors