MATLAB Toolbox **envlp**: User's Guide

June 10, 2014

Contents **1**

1 Overview **2**
- 1.1 Technical Support . . . 3
- 1.2 Directory Structure . . . 3
- 1.3 Document Organization . . . 3

2 Quick Start **6**
- 2.1 Installation . . . 6
- 2.2 A Guided Tour . . . 6
- 2.3 Two examples . . . 8

3 Envelope Models **14**
- 3.1 Multivariate Linear Regression . . . 14
- 3.2 Envelope Model . . . 15
- 3.3 Envelope Model Using Sequential Algorithm . . . 16
- 3.4 Heteroscedastic Envelope Model . . . 17
- 3.5 Inner Envelope Model . . . 19
- 3.6 Partial Envelope Model . . . 20
- 3.7 Scaled Envelope Model . . . 21
- 3.8 Envelope Model in the Predictor Space . . . 23
- 3.9 Envelope Model in the Predictor Space Using Partial Least Squares Algorithm 24
- 3.10 Envelope Estimator for Multivariate Mean . . . 25

4 Optional Arguments **27**

1

Overview

The envelope model is a new area in multivariate analysis. It uses dimension reduction techniques to achieve efficient estimation of parameters, for example, the regression coefficients in multivariate linear regression (MLR).

This MATLAB toolbox **envlp** currently has nine modules: **env**, **envseq**, **henv**, **ienv**, **penv**, **senv**, **xenv**, **xenvpls** and **envmean**. The nine modules implement six models in the envelope family, including envelope model (using Grassmann manifold optimization algorithm and sequential algorithm), heteroscedastic envelope model, inner envelope model, partial envelope model, scaled envelope model and envelope model in the predictor space (using Grassmann manifold optimization algorithm and partial least squares algorithm), as well as the envelope estimator for multivariate mean. The modules are described as follows.

env Implements the envelope model. The envelope model is a general tool for efficient estimation in the context of MLR, and it has the potential to achieve substantial efficiency gains when part of the response variables or their linear combination is invariant to the changes of the predictors [**?**].

envseq Implements the envelope model using sequential algorithm. This module also implements the envelope model, but uses a sequential computing algorithm. This algorithm makes the envelope model applicable for small sample size cases [**?**].

henv Implements the heteroscedastic envelope model. The heteroscedastic envelope model is used when the data has non constant error structure [**?**].

ienv Implements the inner envelope model. The inner envelope model is also a general tool for efficient estimation in the context of MLR, but has a different mechanism from the envelope model. Therefore it may provide efficiency gains in the cases when the envelope models fail to offer any gains [**?**].

penv Implements the partial envelope model. The partial envelope model can be applied when part of the predictors are of main interest. It often gives more efficiency gains than the envelope model in estimating the coefficients of the main predictors [**?**].

senv Implements the scaled envelope model. The scaled envelope model is used when the user hopes to have a scale-invariant version of the envelope model [**?**, 2012].

xenv Implements the envelope model in the predictor space. The envelope model in the predictor space is used when some of the predictors or their linear combinations

do not contribute to the change of the responses. It can potentially bring a better prediction performance than the standard model, or even partial least squares [**?**].

xenvpls Implements the envelope model in the predictor space using partial least squares algorithm. This module implements the envelope model in the predictor space, but uses the partial least squares algorithm. This algorithm makes the envelope model in the predictor space applicable for small sample size cases [**?**].

envmean Implements the envelope estimator of the multivariate mean. The envelope estimator of the multivariate mean has smaller mean squared error and asymptotic standard errors compared to the sample mean, and often has smaller mean squared error compared to the James-Stein estimator [**?**].

The complete applicability of this toolbox is described in Table 1.1.

1.1. Technical Support

We provide a support website, on which the users can download the toolbox, report bugs and check recent updates http://code.google.com/p/envlp/. For further help, the users can contact the authors of the toolbox: Dennis Cook (dennis@stat.umn.edu), Zhihua Su (zhihuasu@stat.ufl.edu), and Yi Yang (yiyang@umn.edu).

1.2. Directory Structure

Toolbox folder contains following files and sub-folders:

./envelope_license.m Envelope Toolbox License.

./install_envelope.m Envelope Toolbox installation script.

./README.txt Envelope Toolbox Quick Start.

./data/ Data published in papers on envelope models and data description manual.

./doc/ User's guide, function reference manual.

./examples/ Examples for demonstration.

./src/ Envelope toolbox source code.

1.3. Document Organization

This user's guide is organized as follows:

Chapter 1 Overview. Introduces the content and applicability of the toolbox.

Chapter 2 Quick start. Describes the installation of the toolbox, and provides simple examples on using the main functions.

Module	Dimension Selection	Inference Tools	Section
env	AIC BIC LRT *m*-fold CV	Estimation and Prediction Bootstrap for Estimating Standard Errors Hypothesis Test on Coefficients	3.2
envseq	*m*-fold CV	Bootstrap for Estimating Standard Errors	3.3
henv	AIC BIC LRT *m*-fold CV	Estimation and Prediction Bootstrap for Estimating Standard Errors Hypothesis Test on Coefficients	3.4
ienv	AIC BIC LRT *m*-fold CV	Estimation and Prediction Bootstrap for Estimating Standard Errors Hypothesis Test on Coefficients	3.5
penv	AIC BIC LRT *m*-fold CV	Estimation and Prediction Bootstrap for Estimating Standard Errors Hypothesis Test on Coefficients	3.6
senv	AIC BIC *m*-fold CV	Estimation and Prediction Bootstrap for Estimating Standard Errors Hypothesis Test on Coefficients	3.7
xenv	AIC BIC LRT *m*-fold CV	Estimation and Prediction Bootstrap for Estimating Standard Errors Hypothesis Test on Coefficients	3.8
xenvpls	*m*-fold CV	Bootstrap for Estimating Standard Errors	3.9
envmean	AIC BIC LRT *m*-fold CV	Estimation and Prediction Bootstrap for Estimating Standard Errors Hypothesis Test on Coefficients	3.10

Table 1.1: Applicability of toolbox envlp

Chapter 3 Envelope models. Discusses each module of the toolbox, and demonstrates the applicability with more detailed examples.

Chapter 4 Optional arguments. Explains how to control the convergence speed, input user-specified starting values, and display iteration process.

This user's guide is intended to assist the users in cognizing and mastering the usage of the toolbox, not all commands are discussed in details as illustrated. For more details, the users can consult the technical Reference.

2

Quick Start

2.1. Installation

To install the toolbox, direct your MATLAB working directory to the folder "envlp", and type

```
install_envlp
```

If the users agree with our license statements, the installation is completed, and all the utilities of the toolbox are added to the MATLAB path. Once installed, no further action is needed to call the functions in the toolbox, even if the users change a working directory, or MATLAB is relaunched. If a previous version of the toolbox is present, it should be removed before installing the new version. To remove, simply delete the folder of the toolbox.

2.2. A Guided Tour

This "envlp" toolbox consists tools in the following three classes:

1. Dimension selection: Select the dimension of the envelope subspace.
2. Model Fitting with Selected Dimension: Fit the model.
3. Post processing: Inference based on the model fitting.

We will present a tour of our toolbox through all three classes.

Dimension selection

The dimension selection tools available in this toolbox are Akaike information criterion (AIC), Bayesian information criterion (BIC), Likelihood ratio testing (LRT) and m-fold cross validation. The m-fold cross validation can be applied to all methods, while AIC, BIC and LRT can be applied to 'env', 'henv', 'ienv', 'penv', 'senv' and 'xenv', except that LRT cannot be applied to 'senv', as indicated in Table 1.1. AIC and BIC generally require longer computing time than LRT, because of the nature of the method. The syntaxes of AIC, BIC, LRT and m-fold cross validation are

```
u = modelselectaic(X, Y, modelType)
u = modelselectbic(X, Y, modelType)
u = modelselectlrt(X, Y, alpha, modelType)
SelectOutput = mfoldcv(X, Y, m, modelType)
```

For the inputs: X is a matrix containing the predictors, except that with penv, it is a list having X.X1 and X.X2; Y is a matrix containing the response; 'alpha' is the significance level; 'm' is an integer indicating m-fold cross validation and 'modelType' is a string indicting the model. The choices for 'modelType' in modelselectaic, modelselectbic and modelselectlrt can be 'env', 'henv', 'ienv', 'penv', 'senv'and 'xenv', and the choices for 'modelType' in mfoldcv are 'envseq'and 'xenvpls'. The output is the dimension of the envelope subspace selected by the tool for modelselectaic, modelselectbic and modelselectlrt, and a list containing dimension of the envelope subspace as well as the prediction error for each dimension for mfoldcv. Examples will be provided in Section 2.3.

Model Fitting with Selected Dimension

The functions to fit the six models in the envelope family are the main drive of this toolbox. Their syntaxes are

```
ModelOutput = env(X, Y, u)
ModelOutput = envseq(X, Y, u)
ModelOutput = henv(X, Y, u)
ModelOutput = ienv(X, Y, u)
ModelOutput = penv(X, Y, u)
ModelOutput = senv(X, Y, u)
ModelOutput = xenv(X, Y, u)
ModelOutput = xenvpls(X, Y, u)
```

The inputs X and Y are the same as the inputs in the dimension selection tools, and u is the dimension of the envelope subspace. The user can specify the dimension, or use dimension selection tools described in dimension selection part to choose u. The output ModelOutput is a list, containing the maximum likelihood estimators (MLE) under the model, as well as some key statistics for inference, including the standard error ratios of elements in the regression coefficients for the standard model versus the envelope model, maximized value of the log likelihood, number of parameters, etc.

envseq and xenvpls fit the same model as env and xenv, respectively, but use a different computing algorithm, making them capable to accommodate small sample size cases. With large sample size cases, envseq and xenvpls can also be applied, and usually give different results from env and xenv. In this case, **?** indicates that env and xenv normally give better results, and therefore are recommended.

Post-processing

This toolbox provides functions for the following inference:

- Compute bootstrap standard errors for the regression coefficients $\widehat{\boldsymbol{\beta}}$, which gives an estimator for the actual standard errors of $\widehat{\boldsymbol{\beta}}$.
- For a given data point, report its fitted value or predicted value, with associated standard errors.
- Test hypothesis

$$H_0: \quad \mathbf{L}\widehat{\boldsymbol{\beta}}\mathbf{R} = \mathbf{A}$$
$$H_\alpha: \quad \mathbf{L}\widehat{\boldsymbol{\beta}}\mathbf{R} \neq \mathbf{A}$$

  where **L**, **R** and **A** are given matrices.

The syntaxes of the functions for the inferences are

```
bootse = bootstrapse(X, Y, u, B, modelType)
PredictOutput = prediction(ModelOutput, Xnew, infType, modelType)
TestOutput = testcoefficient(ModelOutput, modelType, TestInput)
```

respectively. For the inputs, modelType in all three functions can be 'env', 'henv', 'ienv', 'penv', 'senv' and 'xenv', while modelType in bootstrapse can also be 'envseq' and 'xenvpls'; ModelOutput in prediction and testcoefficient is a list returned from the functions for model fitting; X, Y, and u are discussed in the dimension selection part; B is the number of bootstrap sample; Xnew is a value of X with which to estimate or predict Y; infType is a string of characters indicting the inference type, the choices are "estimation" and "prediction"; and TestInput is a list that specifies the null hypothesis quantities **L**, **R**, and **A**. TestInput is an option input argument; if missing, testcoefficient will perform the usual F test, i.e. testing if $\boldsymbol{\beta} = 0$. The output of bootstrapse is a matrix the same size of $\widehat{\boldsymbol{\beta}}$, with each elements as the bootstrap standard error of its corresponding element in $\widehat{\boldsymbol{\beta}}$. The output of prediction is a list containing the fitted or predicted value, its covariance matrix and standard errors. The function testcoefficient will print out a form displaying the test statistic, degrees of freedom of the reference distribution and the p-value, all of which and some other statistics are in the list TestOutput.

For functions in envmean module, the interface is different from those in other modules, since the context is not regression. The functions in envmean module will be explained in details in Section 3.10.

2.3. Two examples

To demonstrate the usage of this toolbox, we take the envelope model and partial envelope model as examples. Other methods can be applied similarly, and are also discussed in details in Chapter 3.

Wheat Protein Data

For the demonstration of the envelope model, we apply it to the wheatprotein data, which is used as a data analysis example in **?**. First we load the data and assign column 1 to 6 as the response Y and column 8 as the predictor X. The description of the data is available under the folder /envlp/data.

```
load wheatprotein.txt
X = wheatprotein(:, 8);
Y = wheatprotein(:, 1:6);
```

Then we apply the dimension selection tools to choose u and get the following results.

```
modelType = 'env';
u = modelselectaic(X, Y, modelType)
u =
     1
u = modelselectbic(X, Y, modelType)
u =
     1

alpha = 0.01;
u = modelselectlrt(X, Y, alpha, modelType)
u =
     1
```

In this example, all three dimension selection tools agree that the dimension of the envelope should be 1, which is consistent with the results in **?**. Now we fit the envelope model with dimension 1.

```
ModelOutput = env(X, Y, u)
ModelOutput =
         beta: [6x1 double]
        Sigma: [6x6 double]
        Gamma: [6x1 double]
       Gamma0: [6x5 double]
          eta: 8.5647
        Omega: 7.8762
       Omega0: [5x5 double]
        alpha: [6x1 double]
            l: -850.7592
    covMatrix: [6x6 double]
        asySE: [6x1 double]
        ratio: [6x1 double]
     paramNum: 28
            n: 50
```

We notice that ModelOutput is a list that includes the MLEs and the statistics relevant to the inference of the envelope model. For more details of the components in ModelOutput, please refer to the Reference of the toolbox. If we want to see the regression coefficients, as well as the standard error ratios, we can type

```
ModelOutput.beta
ans =
   -1.0644
    4.4730
    3.6839
   -5.9770
    0.6013
   -1.5986

ModelOutput.ratio
ans =
   28.0389
   18.3983
   23.5916
   16.2928
   65.7999
    6.4555
```

and get the results. We can also look at the eigenvalues of $\mathbf{\Sigma}_1$ and $\mathbf{\Sigma}_2$ by the following commands

```
ModelOutput.Omega
ans =
    7.8762

eig(ModelOutput.Omega0)
ans =
   1.0e+03 *
    6.5166
    0.2083
    0.0201
    0.0004
    0.0003
```

These results are consistent with those published in **?**. After getting the model fitting results, if we would like to check the fitted value and its standard errors for the second observation, we can use the prediction function.

```
Xnew = X(2, :)';
PredictOutput = predict_env(ModelOutput, Xnew, 'estimation')
PredictOutput =
```

```
      value: [6x1 double]
   covMatrix: [6x6 double]
         SE: [6x1 double]
```

We can check the standard errors of the fitted value, and also compare the fitted value with its true value

```
PredictOutput.SE
ans =
   4.8892
   4.0227
   4.3237
   4.7470
   6.8186
   2.6948

[PredictOutput.value, Y(2,:)']
ans =
  474.7135  458.0000
  127.4740  112.0000
  251.2044  236.0000
  380.8280  368.0000
  380.9473  383.0000
   -6.3287  -15.0000
```

Now suppose we want to test if $\beta = 0$, we run the testcoefficient as

```
TestOutput = testcoefficient(ModelOutput, modelType)
 Test Hypothesis     Chisq Statistic    DF     P-value
-------------------------------------------------------
L * beta * R = A        116.230         6       0.0000
-------------------------------------------------------
```

Notice that we did not have TestInput, because we are performing the usual F test on if $\beta = 0$. The test results indicate that we have strong evidence to reject $\beta = 0$.

Fiber and Paper Data

Now we use another example for the partial envelope model. The data loaded is the fiber and paper data analyzed in **?**.

```
load fiberpaper.dat
Y = fiberpaper(:, 1 : 4);
X.X1 = fiberpaper(:, 7);
X.X2 = fiberpaper(:, 5 : 6);
```

We notice that for partial envelope model, X is a list, with X.X1 containing the main predictors and X.X2 containing covariates. The dimension selection process is parallel to that for the envelope model:

```
modelType = 'penv';
u = modelselectaic(X, Y, modelType)
u =

    3

u = modelselectbic(X, Y, modelType)
u =

    1

alpha = 0.01;
u = modelselectlrt(X, Y, alpha, modelType)
u =

    1
```

In this example, AIC picks the dimension 3, while BIC and LRT both gives dimension as 1. In **?**, $u = 1$ is used for model fitting and inference. So we fit the partial envelope model with $u = 1$.

```
u = 1;
ModelOutput = penv(X, Y, u)
ModelOutput =
       beta1: [4x1 double]
       beta2: [4x2 double]
       alpha: [4x1 double]
       Gamma: [4x1 double]
         eta: 0.0047
       Omega: 0.0149
      Omega0: [3x3 double]
       Sigma: [4x4 double]
           l: -35.6323
    paramNum: 23
   covMatrix: [12x12 double]
      asySE: [4x1 double]
      ratio: [4x1 double]
           n: 62
```

Again we get the output as a list, which contains the MLEs for the parameters and statistics that are relevant to the inference for the partial envelope model. Detailed description of the elements in the list ModelOutput is in the Reference of the toolbox. To display the standard error ratios, and the eigenvalues of $\mathbf{\Sigma}_1$ and $\mathbf{\Sigma}_2$, we key in

```
ModelOutput.ratio
ans =
   65.9692
    6.8217
   10.4152
    9.6228

ModelOutput.Omega
ans =
    0.0149

eig(ModelOutput.Omega0)
ans =
    4.9819
    0.0999
    0.0050
```

The results are almost the same the those in **?**. They are slightly different, because the toolbox uses a different starting value algorithm, which is less likely to be trapped in local minimum and thus leads to more reliable results. Next we look at the standard errors for elements in $\widehat{\boldsymbol{\beta}}$. Since ModelOutput.asySE contains the asymptotic standard errors, for actual standard errors, we need to divide the asymptotic standard errors by $\sqrt{n}$, where n is the sample size and is returned in ModelOutput.n.

```
ModelOutput.asySE / sqrt(ModelOutput.n)
ans =
   1.0e-03 *
    0.3181
    0.8704
    0.9719
    0.4689
```

The standard errors above are computed assuming the partial envelope model. They can also be estimated using bootstrap.

```
B = 5000;
bootse = bootstrapse(X, Y, u, B, modelType)
bootse =
    0.0052
    0.0020
    0.0029
    0.0013
```

The uses may get slightly different results each time they run bootstrapse, as different random seeds are used.

3

Envelope Models

3.1. Multivariate Linear Regression

The envelope model is originally developed under the framework of multivariate linear regression, and its performance is frequently compared to the standard multivariate linear regression model. In this toolbox, we offer some simple functions on multivariate linear regression to ease the comparison between the two models.

A standard multivariate linear regression model is formulated as

$$\mathbf{Y} = \boldsymbol{\alpha} + \boldsymbol{\beta}\mathbf{X} + \boldsymbol{\varepsilon},$$

where $\mathbf{Y} \in \mathbb{R}^r$ is the multivariate response, $\mathbf{X} \in \mathbb{R}^p$ is non-stochastic predictor, and $\boldsymbol{\varepsilon} \in \mathbb{R}^r$ follows a distribution with mean 0, and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{r\times r}$, $\boldsymbol{\alpha} \in \mathbb{R}^r$ and $\boldsymbol{\beta} \in \mathbb{R}^{r\times p}$ are the unknown intercept and coefficients. The goal of the envelope model is to reduce the standard errors in estimating $\boldsymbol{\beta}$.

This function fits the standard multivariate linear regression model

```
ModelOutput = fit_OLS(X, Y)
```

where the input X is an $n \times p$ matrix, with the ith row being the transpose of the ith observation of $\mathbf{X}$, and Y is an $n \times r$ matrix with the ith row being the transpose of the ith observation of $\mathbf{Y}$. The output ModelOutput is a list, which contains the ordinary least squares (OLS) estimators of $\boldsymbol{\beta}$, $\boldsymbol{\Sigma}$ and $\boldsymbol{\alpha}$.

The standard errors of the OLS estimator $\widehat{\boldsymbol{\beta}}$ is often compared to those from the envelope model, the function

```
bootse = bootstrapse_OLS(X, Y, B)
```

computes the estimated standard errors of $\widehat{\boldsymbol{\beta}}$ by bootstrap. The input B is the number of bootstrap samples, and bootse returns a matrix the same size as $\widehat{\boldsymbol{\beta}}$, with each element as the standard error of the corresponding element in $\widehat{\boldsymbol{\beta}}$.

3.2. Envelope Model

The envelope model is a general method in the envelope family to reduce the standard errors in estimating $\boldsymbol{\beta}$. It can be applied when the number of the responses is strictly greater than the number of the predictors, and the responses are continuous variables. It has a potential to obtain efficiency gains in estimating $\boldsymbol{\beta}$, compared to the OLS estimators.

In the multivariate linear regression context in Section 3.1, a coordinate form of the envelope model is

$$\mathbf{Y} = \boldsymbol{\alpha} + \boldsymbol{\Gamma}\boldsymbol{\eta}\mathbf{X} + \boldsymbol{\varepsilon}, \quad \boldsymbol{\Sigma} = \boldsymbol{\Gamma}\boldsymbol{\Omega}\boldsymbol{\Gamma}^T + \boldsymbol{\Gamma}_0\boldsymbol{\Omega}_0\boldsymbol{\Gamma}_0^T,$$

where the regression coefficients $\boldsymbol{\beta} = \boldsymbol{\Gamma}\boldsymbol{\eta}$, $\mathcal{B} = \text{span}(\boldsymbol{\beta})$, $\boldsymbol{\Gamma} \in \mathbb{R}^{r\times u}$ spans $\mathcal{E}_{\boldsymbol{\Sigma}}(\mathcal{B})$ – the envelope subspace, $\boldsymbol{\Gamma}_0 \in \mathbb{R}^{r\times(r-u)}$ spans the orthogonal complement of $\mathcal{E}_{\boldsymbol{\Sigma}}(\mathcal{B})$, $\boldsymbol{\eta} \in \mathbb{R}^{u\times p}$, $\boldsymbol{\Omega} \in \mathbb{R}^{u\times u}$, $\boldsymbol{\Omega}_0 \in \mathbb{R}^{(r-u)\times(r-u)}$ are coordinates, and u is the dimension of $\mathcal{E}_{\boldsymbol{\Sigma}}(\mathcal{B})$.

To select the dimension of the envelope subspace u, we can use the following functions

```
u = modelselectaic(X, Y, 'env')
u = modelselectbic(X, Y, 'env')
alpha = 0.01;  # Users can specify other significance level
u = modelselectlrt(X, Y, alpha, 'env')
```

The possible values of u can be any integer from 0 to r. When $u = r$, the envelope model is equivalent to the standard multivariate linear model. And when $u = 0$, it means that $\boldsymbol{\beta} = 0$, then the changes in $\mathbf{Y}$ do not depend on $\mathbf{X}$. After obtaining u, we can fit the envelope model by

```
ModelOutput = env(X, Y, u)
```

The output ModelOutput is a list, which contains the MLEs of $\boldsymbol{\beta}$, $\boldsymbol{\Sigma}$, $\boldsymbol{\Gamma}$, $\boldsymbol{\Gamma}_0$, $\boldsymbol{\eta}$, $\boldsymbol{\Omega}$, $\boldsymbol{\Omega}_0$ and $\boldsymbol{\alpha}$, and also statistics computed from the model, including the maximized log likelihood, the asymptotic covariance matrix of $\text{vec}(\widehat{\boldsymbol{\beta}})$, the asymptotic standard errors of elements in $\widehat{\boldsymbol{\beta}}$, the ratios of the asymptotic standard errors of the standard model versus the envelope model for elements in $\boldsymbol{\beta}$, the number of parameters in the model, and the number of observations in the data. After fitting the data and get ModelOutput, we can perform post processing inference as computing bootstrap standard errors of $\widehat{\boldsymbol{\beta}}$ by

```
bootse = bootstrapse(X, Y, u, B, 'env')
```

or computing the fitted value or predicted value given an $\mathbf{X}$ by

```
PredictOutput = prediction(ModelOutput, Xnew, 'estimation', 'env')
PredictOutput = prediction(ModelOutput, Xnew, 'prediction', 'env')
```

or testing if some linear combination of $\boldsymbol{\beta}$ is equal to a particular matrix, i.e. given $\mathbf{L}$, $\mathbf{R}$, and $\mathbf{A}$, testing if $\mathbf{L}\boldsymbol{\beta}\mathbf{R} = \mathbf{A}$,

```
TestOutput = testcoefficient(ModelOutput, modelType, TestInput)
```

The inputs and outputs of these post processing functions are discussed in details in Section 2.2.

3.3. Envelope Model Using Sequential Algorithm

The envelope model using sequential algorithm implements the envelope model in Section 3.2, but uses a different computing algorithm. The algorithm estimates the envelope subspace sequentially [**?**]. It is faster than the Grassmann manifold optimization algorithm used in env, and it is applicable when the sample size n is less than the number of responses r. In large sample cases, the performance of this algorithm is not as good as the Grassamn manifold optimization algorithm, but as it is faster, it can provide a starting value for the Grassmann manifold optimization.

To select the dimension of the envelope subspace u, we can use the m-fold cross validation. Common choices are 5 or 10.

```
m = 5;
SelectOutput = mfoldcv(X, Y, m, 'envseq')
```

The possible values of u can be any integer from 0 to the minimum of the floor of $(m-1)n/m - 1$. When $u = 0$, it means that $\boldsymbol{\beta} = 0$, and the changes in $\mathbf{Y}$ do not depend on the changes in $\mathbf{X}$. If the sample size is large, we can also use the tools for 'env' to select u, as env and envseq implement the same model.

```
u = modelselectaic(X, Y, 'env')
u = modelselectbic(X, Y, 'env')
alpha = 0.01;
u = modelselectlrt(X, Y, alpha, 'env')
```

After getting u, to fit the envelope model using the sequential algorithm, we use

```
ModelOutput = envseq(X, Y, u)
```

The output ModelOutput is a list, which contains the estimators of $\boldsymbol{\beta}$, $\boldsymbol{\Sigma}$, $\boldsymbol{\Gamma}$, $\boldsymbol{\Gamma}_0$, $\boldsymbol{\eta}$, $\boldsymbol{\Omega}$, $\boldsymbol{\Omega}_0$ and $\boldsymbol{\alpha}$, and also the number of parameters in the model and number of observations in the data. It does not contain any information based on the likelihood function, for example, the maximized log-likelihood, the asymptotic covariance matrix of $\text{vec}(\widehat{\boldsymbol{\beta}})$, or the asymptotic standard errors of elements in $\widehat{\boldsymbol{\beta}}$. envseq is mainly implemented for small sample size cases, under which the likelihood cannot be estimated. Then any inference that is based on the covariance matrix of $\text{vec}(\boldsymbol{\beta})$ such as estimation, prediction and test coefficients cannot be performed with 'envseq'. But the bootstrap standard errors can still be computed for $\widehat{\boldsymbol{\beta}}$ by

```
bootse = bootstrapse(X, Y, u, B, 'envseq')
```

env can be very slow if r is large, and it cannot be applicable when n is less than r. In those cases, envseq can be used as an alternative. The results of envseq also provide a starting value for env, although it is more often stuck in local minimums than the default starting value of env. Nevertheless, envseq provides a $\sqrt{n}$ consistent estimator of the envelope, then one Gauss-Newton iteration with the output of envseq as the starting value gives an estimator that is asymptotically equivalent to the MLE of $\mathcal{E}_{\boldsymbol{\Sigma}}(\mathcal{B})$. To use the output of envseq as the starting value, an example is shown as follows.

```
load wheatprotein.txt
X = wheatprotein(:, 8);
Y = wheatprotein(:, 1 : 6);
u = 1;
temp = envseq(X, Y, u);
Opts.init = temp.Gamma;
ModelOutput = env(X, Y, u, Opts);
```

The usage of Opts will be discussed in details in Chapter 4.

3.4. Heteroscedastic Envelope Model

The heteroscedastic envelope model [**?**] is used when the data has non constant error structure, and it is developed under the framework of estimating multivariate means for different populations. To use the heteroscedastic envelope model, the number of response should be greater than or equal to the number of populations.

The standard model for estimating multivariate means for p populations can be formulated as $\mathbf{Y}_{(i)j} = \boldsymbol{\mu} + \boldsymbol{\beta}_{(i)} + \boldsymbol{\varepsilon}_{(i)j}$, $i = 1, \cdots, p$, $j = 1, \cdots, n_{(i)}$. We use the subscript (i) to represent the ith group, and we use j without the parenthesis to represent the jth observation. Then $\mathbf{Y}_{(i)j} \in \mathbb{R}^r$ is the jth observation in the ith group, $n_{(i)}$ is the number of observations in the ith group, $\boldsymbol{\mu} \in \mathbb{R}^r$ is the grand mean, $\boldsymbol{\beta}_{(i)} \in \mathbb{R}^r$ is the main effect for the ith group and satisfies $\sum_{i=1}^{p} n_{(i)}\boldsymbol{\beta}_{(i)} = 0$, the errors $\boldsymbol{\varepsilon}_{(i)j}$ follows a distribution with mean 0, and covariance matrix $\boldsymbol{\Sigma}_{(i)} \in \mathbb{R}^{r\times r}$. With this formulation, let $\mathcal{B} = \text{span}(\boldsymbol{\beta}_{(1)} \cdots, \boldsymbol{\beta}_{(p)})$, and

$\mathcal{M} = \{\mathbf{\Sigma}_{(i)} : i = 1, \cdots, p\}$. The coordinate form of the heteroscedastic envelope model is displayed as follows:

$$\mathbf{Y}_{(i)j} = \boldsymbol{\mu} + \mathbf{\Gamma}\boldsymbol{\eta}_{(i)} + \boldsymbol{\varepsilon}_{(i)j}, \quad \mathbf{\Sigma}_{(i)} = \mathbf{\Gamma}\mathbf{\Omega}_{1(i)}\mathbf{\Gamma}^T + \mathbf{\Gamma}_0\mathbf{\Omega}_0\mathbf{\Gamma}_0^T, \quad i = 1, \cdots, p, \ j = 1, \cdots, n_{(i)},$$

where $\boldsymbol{\beta}_{(i)} = \mathbf{\Gamma}\boldsymbol{\eta}_{(i)}$, $\mathbf{\Gamma} \in \mathbb{R}^{r\times u}$ spans $\mathcal{E}_{\mathcal{M}}(\mathcal{B})$ – the $\mathcal{M}$ envelope of $\mathcal{B}$, $\mathbf{\Gamma}_0 \in \mathbb{R}^{r\times(r-u)}$ spans the orthogonal complement of $\mathcal{E}_{\mathcal{M}}(\mathcal{B})$, $\boldsymbol{\eta}_{(i)} \in \mathbb{R}^{u\times 1}$, $\mathbf{\Omega}_{1(i)} \in \mathbb{R}^{u\times u}$, $\mathbf{\Omega}_0 \in \mathbb{R}^{(r-u)\times(r-u)}$, $i = 1, \cdots, p$, are coordinates, $\sum_{i=1}^{p} n_{(i)}\boldsymbol{\eta}_{(i)} = 0$, and u is the dimension of $\mathcal{E}_{\mathcal{M}}(\mathcal{B})$.

To use the heteroscedastic envelope model, first we check if the data has heteroscedastic error structure by the Box's M test [**?**]. Using the water strider example [**?**], the following codes performs the Box's M test.

```
load waterstrider.mat
alpha = 0.01;  # Users can specify other significance level
TestOutput = mtest(X, Y, alpha);
```

The input X is a group indicator, there is no constraint on the number of columns of X, as long as X takes p unique values for p populations. For example, if there are three groups, X can take three unique values as 1, 2, and 3 to indicate the groups, or X can take $(0, 1)$, $(1, 0)$, and $(0, 0)$ to indicate the groups. The output of the M test is displayed below

```
-----------------------------------------------
  MBox     Chi-sqr.        df        P
-----------------------------------------------
157.5977   137.3361        72      0.0000
-----------------------------------------------
Covariance matrices are significantly different.
```

As the data has non constant covariance structure, we need to apply the heteroscedastic envelope model. To select the dimension of the envelope subspace, we use

```
u = modelselectaic(X, Y, 'henv')
u = modelselectbic(X, Y, 'henv')
alpha = 0.01;
u = modelselectlrt(X, Y, alpha, 'henv')
```

The input X has the same form as in the function mtest. After obtaining u, the heteroscedastic envelope model can be fitted by

```
ModelOutput = henv(X, Y, u)
```

The output ModelOutput is a list containing the fitted values $\widehat{\mathbf{Y}}$, the unique values in $\mathbf{X}$, the MLEs of the grand mean $\boldsymbol{\mu}$, the group mean $\boldsymbol{\mu} + \boldsymbol{\beta}_{(i)}$, $\boldsymbol{\Gamma}$, $\boldsymbol{\Gamma}_0$, $\boldsymbol{\beta}_{(i)}$, $\boldsymbol{\Sigma}_{(i)}$, $\boldsymbol{\eta}_{(i)}$, $\boldsymbol{\Omega}_{(i)}$, and $\boldsymbol{\Omega}_0$, and also statistics computed from the model, including the maximized log likelihood, the asymptotic covariance matrix of $(\hat{\boldsymbol{\mu}}^T, \widehat{\boldsymbol{\beta}}_{(1)}^T, \cdots, \widehat{\boldsymbol{\beta}}_{(p)}^T)^T$, the asymptotic standard errors of elements in $\widehat{\boldsymbol{\beta}}_{(i)}$, the ratios of the asymptotic standard errors of the standard model versus the heteroscedastic envelope model for elements in $\boldsymbol{\beta}_{(i)}$, the number of parameters in the model, and the number of observations in the data. After model fitting, the functions for post processing inference are

```
bootse = bootstrapse(X, Y, u, B, 'henv')
PredictOutput = prediction(ModelOutput, Xnew, 'estimation', 'henv')
PredictOutput = prediction(ModelOutput, Xnew, 'prediction', 'henv')
TestOutput = testcoefficient(ModelOutput, modelType, TestInput)
```

Their inputs and outputs are similar to those for the envelope model, except that X here is a group indicator, and Xnew in prediction should be one of the unique value of X in the original data.

3.5. Inner Envelope Model

Like the envelope model, the inner envelope model **[?]** is also a general method to reduce the standard errors in estimating $\boldsymbol{\beta}$. It has a different mechanism to achieve efficiency gains than the envelope model, so it may be very useful when the envelope model reduces to the standard model and offers no efficiency gains. The inner envelope model also requires that the number of predictors is strictly less than the number of responses.

In the multivariate linear regression context (Section 3.1), the coordinate form of the inner envelope model can be written as

$$\mathbf{Y} = \boldsymbol{\alpha} + (\boldsymbol{\Gamma}_1\boldsymbol{\eta}_1^T + \boldsymbol{\Gamma}_0\mathbf{B}\boldsymbol{\eta}_2^T)\mathbf{X} + \boldsymbol{\varepsilon}, \quad \boldsymbol{\Sigma} = \boldsymbol{\Gamma}_1\boldsymbol{\Omega}_1\boldsymbol{\Gamma}_1^T + \boldsymbol{\Gamma}_0\boldsymbol{\Omega}_0\boldsymbol{\Gamma}_0^T,$$

where $\boldsymbol{\beta} = \boldsymbol{\Gamma}_1\boldsymbol{\eta}_1^T + \boldsymbol{\Gamma}_0\mathbf{B}\boldsymbol{\eta}_2^T \in \mathbb{R}^{r\times p}$, $\mathcal{B} = \text{span}(\boldsymbol{\beta})$, $\boldsymbol{\Gamma}_1 \in \mathbb{R}^{r\times u}$ spans the inner envelope subspace $\mathcal{IE}_{\boldsymbol{\Sigma}}(\mathcal{B})$, $\boldsymbol{\Gamma}_0 \in \mathbb{R}^{r\times(r-u)}$ spans the orthogonal complement of $\mathcal{IE}_{\boldsymbol{\Sigma}}(\mathcal{B})$, $\mathbf{B} \in \mathbb{R}^{(r-u)\times(p-u)}$ is a semi-orthogonal matrix such that $(\boldsymbol{\Gamma}_1, \boldsymbol{\Gamma}_0\mathbf{B})$ spans $\mathcal{B}$, $\boldsymbol{\eta}_1 \in \mathbb{R}^{p\times u}$, $\boldsymbol{\eta}_2 \in \mathbb{R}^{p\times(p-u)}$, $\boldsymbol{\Omega}_1 \in \mathbb{R}^{u\times u}$, and $\boldsymbol{\Omega}_0 \in \mathbb{R}^{(r-u)\times(r-u)}$ contain the coordinates, and u is the dimension of $\mathcal{IE}_{\boldsymbol{\Sigma}}(\mathcal{B})$.

To select the dimension of $\mathcal{IE}_{\boldsymbol{\Sigma}}(\mathcal{B})$, we use the same functions as for the envelope model, except that the modelType is 'ienv'.

```
u = modelselectaic(X, Y, 'ienv')
u = modelselectbic(X, Y, 'ienv')
alpha = 0.01;
u = modelselectlrt(X, Y, alpha, 'ienv')
```

The possible values of u are integers from 0 to p. When $u = 0$, the inner envelope model reduces to the standard model, and offers no efficiency gains. When $u = p$, the inner envelope model is equivalent to an envelope model with dimension p.

Given the dimension u, the inner envelope model can be fitted by

```
ModelOutput = ienv(X, Y, u)
```

The output ModelOutput is a list containing the MLEs of $\boldsymbol{\beta}$, $\boldsymbol{\Sigma}$, $\boldsymbol{\Gamma}_1$, $\boldsymbol{\Gamma}_0$, $\boldsymbol{\eta}_1$, $\mathbf{B}$, $\boldsymbol{\eta}_2$, $\boldsymbol{\Omega}_1$, $\boldsymbol{\Omega}_0$ and $\boldsymbol{\alpha}$, and also statistics computed from the model, including the maximized log likelihood, the asymptotic covariance matrix of $\text{vec}(\widehat{\boldsymbol{\beta}})$, the asymptotic standard errors of elements in $\widehat{\boldsymbol{\beta}}$, the ratios of the asymptotic standard errors of the standard model versus the inner envelope model for elements in $\boldsymbol{\beta}$, the number of parameters in the model, and the number of observations in the data. After model fitting, the functions for post processing inference are

```
bootse = bootstrapse(X, Y, u, B, 'ienv')
PredictOutput = prediction(ModelOutput, Xnew, 'estimation', 'ienv')
PredictOutput = prediction(ModelOutput, Xnew, 'prediction', 'ienv')
TestOutput = testcoefficient(ModelOutput, modelType, TestInput)
```

Their inputs and outputs are similar to those for the envelope model.

3.6. Partial Envelope Model

The partial envelope model [**?**] can be applied when part of the predictors are of main interest. It only requires that the number of the main predictors is strictly less than the number of the responses. This is particular useful when the number of the predictors p is large, but only a small number of predictors are of main interest. Suppose that $\mathbf{X}^T = (\mathbf{X}_1^T, \mathbf{X}_2^T)$, where $\mathbf{X}_1 \in \mathbb{R}^{p_1}$ are predictors of main interest, and $\mathbf{X}_2 \in \mathbb{R}^{p_2}$ are covariates, $p_1 + p_2 = p$. Then the standard model is formulated as $\mathbf{Y} = \boldsymbol{\alpha} + \boldsymbol{\beta}_1\mathbf{X}_1 + \boldsymbol{\beta}_2\mathbf{X}_2 + \boldsymbol{\varepsilon}$, and the coordinate form of the partial envelope model is

$$\mathbf{Y} = \boldsymbol{\alpha} + \boldsymbol{\Gamma}\boldsymbol{\eta}\mathbf{X}_1 + \boldsymbol{\beta}_2\mathbf{X}_2 + \boldsymbol{\varepsilon}, \quad \boldsymbol{\Sigma} = \boldsymbol{\Gamma}\boldsymbol{\Omega}\boldsymbol{\Gamma}^T + \boldsymbol{\Gamma}_0\boldsymbol{\Omega}_0\boldsymbol{\Gamma}_0^T,$$

where $\boldsymbol{\beta}_1 = \boldsymbol{\Gamma}\boldsymbol{\eta} \in \mathbb{R}^{r\times p_1}$, $\mathcal{B}_1 = \text{span}(\boldsymbol{\beta}_1)$, $\boldsymbol{\Gamma} \in \mathbb{R}^{r\times u}$ spans the partial envelope subspace $\mathcal{E}_{\boldsymbol{\Sigma}}(\mathcal{B}_1)$, $\boldsymbol{\Gamma}_0 \in \mathbb{R}^{r\times u}$ spans the orthogonal complement of $\mathcal{E}_{\boldsymbol{\Sigma}}(\mathcal{B}_1)$, $\boldsymbol{\eta} \in \mathbb{R}^{u\times p_1}$, $\boldsymbol{\Omega} \in \mathbb{R}^{u\times u}$, $\boldsymbol{\Omega}_0 \in \mathbb{R}^{(r-u)\times(r-u)}$ are coordinates, $\boldsymbol{\beta}_2 \in \mathbb{R}^{r\times p_2}$ contains the coefficients for $\mathbf{X}_2$ and u is the dimension of $\mathcal{E}_{\boldsymbol{\Sigma}}(\mathcal{B}_1)$.

For functions related to the partial envelope model, the input X is a list, which has two elements X.X1 and X.X2. The element X.X1 is an $n \times p_1$ matrix, with the ith row as the transpose of the ith observation of $\mathbf{X}_1$, and X.X2 is an $n \times p_2$ matrix, with the ith row as the transpose of the ith observation of $\mathbf{X}_2$. The form of X as a list is unique to the partial

envelope model. The name of the list can be different, but the names of components in the list are fixed. For example, we can have ABC.X1 and ABC.X2, but we cannot have ABC.Z1 and ABC.Z2.

The following functions can be applied to select u,

```
u = modelselectaic(X, Y, 'penv')
u = modelselectbic(X, Y, 'penv')
alpha = 0.01;
u = modelselectlrt(X, Y, alpha, 'penv')
```

The possible values of u are any integer from 0 to r, when $u = r$, the partial envelope model reduces the standard model, and when $u = 0$, the changes in $\mathbf{Y}$ do not depend on the changes in $\mathbf{X}_1$ given $\mathbf{X}_2$.

After obtaining u, the partial envelope model can be fit by

```
ModelOutput = penv(X, Y, u)
```

The output ModelOutput is a list containing the MLEs of $\boldsymbol{\beta}_1$, $\boldsymbol{\beta}_2$, $\boldsymbol{\Sigma}$, $\boldsymbol{\Gamma}$, $\boldsymbol{\Gamma}_0$, $\boldsymbol{\eta}$, $\boldsymbol{\Omega}$, $\boldsymbol{\Omega}_0$ and $\boldsymbol{\alpha}$, as well as statistics computed from the model, including the maximized log likelihood, the asymptotic covariance matrix of $(\text{vec}(\widehat{\boldsymbol{\beta}}_2)^T, \text{vec}(\widehat{\boldsymbol{\beta}}_1)^T)^T$, the asymptotic standard errors of elements in $\widehat{\boldsymbol{\beta}}_1$, the ratios of the asymptotic standard errors of the standard model versus the partial envelope model for elements in $\boldsymbol{\beta}_1$, the number of parameters in the model, and the number of observations in the data. The functions for post processing inference are

```
bootse = bootstrapse(X, Y, u, B, 'penv')
PredictOutput = prediction(ModelOutput, Xnew, 'estimation', 'penv')
PredictOutput = prediction(ModelOutput, Xnew, 'prediction', 'penv')
TestOutput = testcoefficient(ModelOutput, modelType, TestInput)
```

The usage of these functions is introduced in Section 2.2 and demonstrated to some extent by the Fiber and Paper example in Section 2.3. In the function prediction, the input Xnew is a list, which has Xnew.X1 and Xnew.X2. Xnew.X1 is a p_1 dimensional column vector containing the new value of $\mathbf{X}_1$, and Xnew.X2 is a p_2 dimensional column vector containing the new value of $\mathbf{X}_2$.

3.7. Scaled Envelope Model

Scaled envelope model [**?**] is used when the user hopes to have a scale-invariant version of the envelope model. The scaled envelope estimator is more efficient or at least as efficient as the standard estimator, and sometimes can be more efficient than the envelope estimator,

especially when the envelope subspace has dimension r. However, the scaled envelope estimator normally takes longer to compute because of the nature of its iterative computing algorithm. But users can print out and monitor the model fitting process, which will be discussed in Chapter 4.

In the multivariate linear regression context (3.1), the coordinate form of the scaled envelope model can be written as

$$\mathbf{Y} = \boldsymbol{\alpha} + \boldsymbol{\Lambda}\boldsymbol{\Gamma}\boldsymbol{\eta}\mathbf{X} + \boldsymbol{\varepsilon}, \quad \boldsymbol{\Sigma} = \boldsymbol{\Lambda}\boldsymbol{\Gamma}\boldsymbol{\Omega}\boldsymbol{\Gamma}^T\boldsymbol{\Lambda} + \boldsymbol{\Lambda}\boldsymbol{\Gamma}_0\boldsymbol{\Omega}_0\boldsymbol{\Gamma}_0^T\boldsymbol{\Lambda},$$

where $\boldsymbol{\beta} = \boldsymbol{\Lambda}\boldsymbol{\Gamma}\boldsymbol{\eta} \in \mathbb{R}^{r\times p}$, $\boldsymbol{\Lambda} \in \mathbb{R}^{r\times r}$ is the scaling matrix, it is a diagonal matrix with the first element as 1, and the other diagonal elements as positive real numbers, $\boldsymbol{\Gamma} \in \mathbb{R}^{r\times p}$ spans the envelope subspace $\mathcal{E}_{\boldsymbol{\Lambda}^{-1}\boldsymbol{\Sigma}\boldsymbol{\Lambda}^{-1}}(\boldsymbol{\Lambda}^{-1}\mathcal{B})$, $\boldsymbol{\Gamma}_0 \in \mathbb{R}^{r\times(r-u)}$ spans the orthogonal complement of $\mathcal{E}_{\boldsymbol{\Lambda}^{-1}\boldsymbol{\Sigma}\boldsymbol{\Lambda}^{-1}}(\boldsymbol{\Lambda}^{-1}\mathcal{B})$, $\boldsymbol{\eta} \in \mathbb{R}^{u\times p}$, $\boldsymbol{\Omega} \in \mathbb{R}^{u\times u}$, and $\boldsymbol{\Omega}_0 \in \mathbb{R}^{(r-u)\times(r-u)}$ carry the coordinates, and u is the dimension of the envelope subspace $\mathcal{E}_{\boldsymbol{\Lambda}^{-1}\boldsymbol{\Sigma}\boldsymbol{\Lambda}^{-1}}(\boldsymbol{\Lambda}^{-1}\mathcal{B})$. If $\boldsymbol{\Lambda} = \mathbf{I}_r$, then the scaled envelope model is equivalent to the envelope model.

To select the dimension of $\mathcal{E}_{\boldsymbol{\Lambda}^{-1}\boldsymbol{\Sigma}\boldsymbol{\Lambda}^{-1}}(\boldsymbol{\Lambda}^{-1}\mathcal{B})$, we can apply the following two functions

```
u = modelselectaic(X, Y, 'senv')
u = modelselectbic(X, Y, 'senv')
```

Notice that likelihood ratio testing can not be applied to the model selection of the scaled envelope model, only AIC and BIC can be used to select u. The possible values of u can be any integer from 0 to r. When $u = r$, the scaled envelope model is the same as the standard multivariate linear model. And when $u = 0$, it means that $\boldsymbol{\beta} = 0$, then the changes in $\mathbf{Y}$ do not depend on $\mathbf{X}$. After obtaining u, we can fit the scaled envelope model by

```
ModelOutput = senv(X, Y, u)
```

The output ModelOutput is a list, which contains the MLEs of $\boldsymbol{\beta}$, $\boldsymbol{\Sigma}$, $\boldsymbol{\Lambda}$, $\boldsymbol{\Gamma}$, $\boldsymbol{\Gamma}_0$, $\boldsymbol{\eta}$, $\boldsymbol{\Omega}$, $\boldsymbol{\Omega}_0$ and $\boldsymbol{\alpha}$, and also statistics computed from the model, including the maximized log likelihood, the asymptotic covariance matrix of $\text{vec}(\widehat{\boldsymbol{\beta}})$, the asymptotic standard errors of elements in $\widehat{\boldsymbol{\beta}}$, the ratios of the asymptotic standard errors of the standard model versus the scaled envelope model for elements in $\boldsymbol{\beta}$, the number of parameters in the model, and the number of observations in the data. After model fitting, the users can perform the following post processing inference:

```
bootse = bootstrapse(X, Y, u, B, 'senv')
PredictOutput = prediction(ModelOutput, Xnew, 'estimation', 'senv')
PredictOutput = prediction(ModelOutput, Xnew, 'prediction', 'senv')
TestOutput = testcoefficient(ModelOutput, modelType, TestInput)
```

These functions are used similarly as those for the envelope model.

3.8. Envelope Model in the Predictor Space

The envelope model in the predictor space is used when the number of the predictors is strictly larger than the number of the responses. It has the potential to have better prediction performance compared to the standard model, or even the partial least squares. In fact, in the population version, the envelope estimator in the predictor space is equivalent to the partial least squares estimator, but in the sample version, its performance is normally superior to the partial least squares estimator.

We slightly change the formulation of the standard model to $\mathbf{Y} = \boldsymbol{\mu} + \boldsymbol{\beta}^T\mathbf{X} + \boldsymbol{\varepsilon}$, to be consistent with the notations in **?**. Then the coordinate form of the envelope model in the predictor space is as follows:

$$\mathbf{Y} = \boldsymbol{\mu} + \boldsymbol{\eta}^T\boldsymbol{\Omega}^{-1}\boldsymbol{\Gamma}^T\mathbf{X} + \boldsymbol{\varepsilon}, \quad \boldsymbol{\Sigma}_\mathbf{X} = \boldsymbol{\Gamma}\boldsymbol{\Omega}\boldsymbol{\Gamma}^T + \boldsymbol{\Gamma}_0\boldsymbol{\Omega}_0\boldsymbol{\Gamma}_0^T,$$

where $\boldsymbol{\mu} \in \mathbb{R}^r$ is the intercept, $\boldsymbol{\beta} = \boldsymbol{\Gamma}\boldsymbol{\Omega}^{-1}\boldsymbol{\eta} \in \mathbb{R}^{p\times r}$, $\boldsymbol{\Gamma} \in \mathbb{R}^{p\times u}$ spans the envelope subspace $\mathcal{E}_{\boldsymbol{\Sigma}_\mathbf{X}}(\mathcal{B})$, and $\mathcal{B} = \text{span}(\boldsymbol{\beta}^T)$, $\boldsymbol{\Gamma}_0 \in \mathbb{R}^{p\times(p-u)}$ spans the orthogonal complement of $\mathcal{E}_{\boldsymbol{\Sigma}_\mathbf{X}}(\mathcal{B})$, $\boldsymbol{\eta} \in \mathbb{R}^{u\times r}$, $\boldsymbol{\Omega} \in \mathbb{R}^{u\times u}$, and $\boldsymbol{\Omega}_0 \in \mathbb{R}^{(p-u)\times(p-u)}$ carry coordinates, and u is the dimension of the envelope $\mathcal{E}_{\boldsymbol{\Sigma}_\mathbf{X}}(\mathcal{B})$.

To select the dimension of $\mathcal{E}_{\boldsymbol{\Sigma}_\mathbf{X}}(\mathcal{B})$, we apply the following three functions

```
u = modelselectaic(X, Y, 'xenv')
u = modelselectbic(X, Y, 'xenv')
alpha = 0.01;
u = modelselectlrt(X, Y, alpha, 'xenv')
```

The possible values for u can be any integer from 0 to p, when $u = p$, the envelope model reduces to the standard model, and when $u = 0$, the changes in $\mathbf{Y}$ do not depend on $\mathbf{X}$. After estimating u, we can fit the model by

```
ModelOutput = xenv(X, Y, u)
```

The output ModelOutput is a list, which contains the MLEs of $\boldsymbol{\beta}$, $\boldsymbol{\Sigma}_\mathbf{X}$, $\boldsymbol{\Gamma}$, $\boldsymbol{\Gamma}_0$, $\boldsymbol{\eta}$, $\boldsymbol{\Omega}$, $\boldsymbol{\Omega}_0$ and $\boldsymbol{\mu}$, and also statistics computed from the model, including the maximized log likelihood, the asymptotic covariance matrix of $\text{vec}(\widehat{\boldsymbol{\beta}})$, the asymptotic standard errors of elements in $\widehat{\boldsymbol{\beta}}$, the ratios of the asymptotic standard errors of the standard model versus the envelope model for elements in $\boldsymbol{\beta}$, the number of parameters in the model, and the number of observations in the data. After model fitting, the following post processing inference can be performed:

```
bootse = bootstrapse(X, Y, u, B, 'xenv')
PredictOutput = prediction(ModelOutput, Xnew, 'estimation', 'xenv')
PredictOutput = prediction(ModelOutput, Xnew, 'prediction', 'xenv')
TestOutput = testcoefficient(ModelOutput, modelType, TestInput)
```

These functions are used similarly as those for the envelope model in Section 3.2.

3.9. Envelope Model in the Predictor Space Using Partial Least Squares Algorithm

The envelope model in the predictor space using the partial least squares algorithm estimates exactly the same model as in section 3.8, but it estimates the envelope subspace $\mathcal{E}_{\mathbf{\Sigma_X}}(\mathcal{B})$ using partial least squares. This makes the envelope model in the predictor space faster to compute, and applicable to small sample cases, where n is less than p.

To select u, the dimension of $\mathcal{E}_{\mathbf{\Sigma_X}}(\mathcal{B})$, we can use m-fold cross validation, for example, 5-fold cross validation:

```
SelectOutput = mfoldcv(X, Y, m, 'xenvpls')
```

The possible values of u can be any integer from 0 to the minimum of $(m-1)n/m - 1$. When $u = p$, the envelope model degenerates to the standard model, and when $u = 0$, $\boldsymbol{\beta} = 0$, and the changes in Y do not depend the changes in $\mathbf{X}$. If the sample size is large, we can also use the dimension selection tools for 'xenv', as xenv and xenvpls implement the same model.

```
u = modelselectaic(X, Y, 'xenv')
u = modelselectbic(X, Y, 'xenv')
alpha = 0.01;
u = modelselectlrt(X, Y, alpha, 'xenv')
```

After obtaining u, to fit the envelope model in the predictor space using partial least squares algorithm, we call the following function

```
ModelOutput = xenvpls(X, Y, u)
```

The output ModelOutput is a list, which contains the estimators of $\boldsymbol{\beta}$, $\mathbf{\Sigma_X}$, $\mathbf{\Gamma}$, $\mathbf{\Gamma}_0$, $\boldsymbol{\eta}$, $\mathbf{\Omega}$, $\mathbf{\Omega}_0$ and $\boldsymbol{\mu}$, and also the number of parameters in the model and number of observations in the data. It does not contain any information based on the likelihood function, for example, the maximized log-likelihood, the asymptotic covariance matrix of $\text{vec}(\widehat{\boldsymbol{\beta}})$, the asymptotic standard errors of elements in $\boldsymbol{\beta}$, etc. xenvpls is mainly implemented for small sample size cases, under which the likelihood cannot be estimated. The estimator of $\mathcal{E}_{\mathbf{\Sigma_X}}(\mathcal{B})$ is obtained by partial least squares algorithm. We can check this by

```
ModelOutput = xenvpls(X, Y, u);
[XL, YL, XS, YS, BETA, PCTVAR, MSE, stats] = plsregress(X, Y, u);
subspace(ModelOutput.Gamma, stats.W)
```

The function plsregress is built in the **Statistics** toolbox in MATLAB, it does the partial least squares regression to the data. The function subspace computes the largest angle between the two subspaces, if the angle is 0, then the two subspaces are the same.

For inference, any inference that is based on the covariance matrix of $\text{vec}(\widehat{\boldsymbol{\beta}})$ such as estimation, prediction and test coefficients cannot be performed with 'xenvpls'. But we can still estimate the bootstrap standard errors for $\widehat{\boldsymbol{\beta}}$ by

```
bootse = bootstrapse(X, Y, u, B, 'xenvpls')
```

3.10. Envelope Estimator for Multivariate Mean

When estimating the multivariate mean from the data, the envelope estimator has a smaller mean squared error than the sample mean, and often has a smaller mean squared error than James-Stein estimator [**?**].

The coordinate form of the envelope model for estimating the multivariate mean is

$$\boldsymbol{\mu} = \boldsymbol{\Gamma}\boldsymbol{\eta} + \boldsymbol{\varepsilon}, \quad \boldsymbol{\Sigma} = \boldsymbol{\Gamma}\boldsymbol{\Omega}\boldsymbol{\Gamma}^T + \boldsymbol{\Gamma}_0\boldsymbol{\Omega}_0\boldsymbol{\Gamma}_0^T,$$

where $\boldsymbol{\Gamma} \in \mathbb{R}^{p\times u}$ spans the envelope subspace $\mathcal{E}_{\boldsymbol{\Sigma}}(\mathcal{M})$, $\mathcal{M} = \text{span}(\boldsymbol{\mu})$, $\boldsymbol{\Gamma}_0 \in \mathbb{R}^{p\times(p-u)}$ spans the orthogonal complement of $\mathcal{E}_{\boldsymbol{\Sigma}}(\mathcal{M})$, $\boldsymbol{\eta} \in \mathbb{R}^u$, $\boldsymbol{\Omega} \in \mathbb{R}^{u\times u}$, $\boldsymbol{\Omega}_0 \in \mathbb{R}^{(p-u)\times(p-u)}$ carry coordinates, and $0 \le u \le p$ is the dimension of the envelope subspace.

Because it is not in the regression setting, the interface of the functions is different from other envelope models. To select the dimension of the envelope subspace u, we can use the following functions

```
u = aic_envmean(Y)
u = bic_envmean(Y)
alpha = 0.01;  # Users can specify other significance level
u = lrt_envmean(Y, alpha)
```

For the input, Y is an $n \times p$ data matrix, where the ith row is $\mathbf{Y}_i^T$, alpha is the significance level, which is often taken as 0.01 or 0.05. The possible values of u can be any integer from 0 to p. When $u = p$, the envelope estimator reduces to the sample mean, $\hat{\boldsymbol{\mu}}_{EM} = \bar{\mathbf{Y}}$. And when $u = 0$, it means that $\boldsymbol{\mu}$ is inferred to be a zero vector. After obtaining u, we can get the estimator of the multivariate mean by

```
ModelOutput = envmean(Y, u)
```

The user can specify the dimension u, or use the dimension selection tools discussed above to choose u. The output ModelOutput is a list, which contains the MLEs of $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$, $\boldsymbol{\Gamma}$, $\boldsymbol{\Gamma}_0$, $\boldsymbol{\eta}$, $\boldsymbol{\Omega}$ and $\boldsymbol{\Omega}_0$, and also statistics computed from the model, including the maximized log likelihood, the asymptotic covariance matrix of $\hat{\boldsymbol{\mu}}$, the asymptotic standard errors of elements in $\hat{\boldsymbol{\mu}}$, the element-wise ratio of the asymptotic standard errors in the sample mean versus the envelope estimator, the number of parameters in the model, and the number of observations in the data. After fitting the data and get ModelOutput, we can perform post processing inference as computing bootstrap standard errors of $\hat{\boldsymbol{\mu}}$ by

```
bootse = bstrp_envmean(Y, u, B)
```

or computing predicted value for a new observation by

```
PredictOutput = predict_envmean(ModelOutput, 'prediction')
```

or testing if some linear combination of $\boldsymbol{\mu}$ is equal to a particular vector, i.e. given $\mathbf{L}$ and $\mathbf{A}$, testing if $\mathbf{L}\boldsymbol{\beta} = \mathbf{A}$,

```
TestOutput = testcoefficient_envmean(ModelOutput, TestInput)
```

In bstrp_envmean, the input B is the number of bootstrap sample, the output bootse is a p dimensional vector with each element as the standard error for the corresponding element in $\hat{\boldsymbol{\mu}}$. The output of predict_envmean is a list containing the predicted value for a new observation, its prediction covariance matrix and standard errors of its elements. For testcoefficient_envmean, the input TestInput is a list containing two components TestInput.L and TestInput.A. TestInput.L is a $d1 \times p$ matrix, with $1 \leq d1 \leq p$, and TestInput.A is a $d1$ column dimensional vector. If TestInput is missing, the default values are used, with TestInput.L being a p by p identity matrix and TestInput.A being a p dimensional zero vector, and the test is if $\mu = 0$. The output TestOutput is a list which returns the chi-squared statistic, degrees of freedom of the statistic, the p-value of the test and the covariance matrix of $\mathbf{L}\boldsymbol{\beta}$. A table will also be printed out to display the test results.

4

Optional Arguments

The computing of the envelope models involves Grassmann manifold optimization. The package sg_min 2.4.3 by Ross Lippert (http://web.mit.edu/~ripper/www/sgmin.html) uses analytical first derivative and numerical second derivative of the objective function, and we find it very stable. The Grassmann manifold optimization in this toolbox is then based on sg_min 2.4.3. Because of the iterative nature of the optimization, we offer some optional arguments so that the users can control the convergence tolerance and speed, as well as monitoring the iteration process. In the functions modelselectaic, modelselectbic, modelselectlrt, env, henv, ienv, penv, senv, xenv, envmean and bootstrapse, there is an optional input argument Opts. Opts is a list, which has five elements:

1. Opts.maxIter: This controls the maximum number of iterations. The default value is 300. The iterations will terminate once the maximum number of iterations is reached, then the results are based on the last iteration. At the same time, a warning

   ```
   WARNING: reached maximum number of iterations without convergence
   for specified tolerances
   ```

   is printed out.

2. Opts.ftol: This controls the tolerance parameter of the objective function. The default value is $1e-10$. The iteration will terminate once the tolerance conditions for both the objective function and its derivative are reached.

3. Opts.gradtol: This controls the tolerance parameter of the derivative of the objective function. The default value is $1e-7$.

4. Opts.verbose: This is a flag for whether or not to print out the iteration process. It is logical 0 or 1, with 0 for no print out and 1 for print out. The default value is 0. The print out will depend on the nature of the functions: with functions for dimension selection (modelselectaic, modelselectbic and modelselectlrt), the current dimension will be printed out; with functions for bootstrap (bootstrapse), the current number of bootstrap sample will be printed out; with the scaled envelope model (senv), the current number of iterations for the alternating algorithm between the scales and Grassmann manifold optimization is printed out; and with all the other functions henv, ienv, penv, senv, xenv and envmean, the current number of iterations of the

Grassmann manifold optimization, -2 times the log likelihood function as well as its gradient are printed out.

5. Opts.init: This argument allows the users to input their starting values. If not specified, the starting values are generated by the functions get_Init or get_Init4henv. This argument is only applicable to env, henv, ienv, penv, senv, xenv and envmean.

The users can choose to define some or none of the five elements, if not defined, the default value will be used. If the users choose to define some of the elements, it is added as the last input of the function. For example, suppose the user defines Opts for env, then the syntax of env will be

```
ModelOutput = env(X, Y, u, Opts)
```

Next we will demonstrate how to use the optional arguments to control convergence and control the display.

Control Convergence Arguments

We use the wheat protein data and the function env as an example. First we set

```
Opts.verbose = 1;
```

so that we can monitor the iteration process. We leave the other arguments as default values for now and fit the envelope model

```
load wheatprotein.txt
X = wheatprotein(:, 8);
Y = wheatprotein(:, 1:6);
u = 1;
ModelOutput = env(X, Y, u, Opts);
iter        grad                F(Y)
0       1.554710e+02        1.702843e+03
1       1.169893e+01        1.701527e+03
2       1.520058e+01        1.701521e+03
3       2.180808e+00        1.701519e+03
4       3.006970e+00        1.701519e+03
5       9.535630e-01        1.701519e+03
6       1.932848e+00        1.701519e+03
7       6.326229e-01        1.701519e+03
8       5.210354e-01        1.701519e+03
9       5.599613e-01        1.701519e+03
10        3.862568e-01        1.701518e+03
11        2.326454e-03        1.701518e+03
12        1.786087e-03        1.701518e+03
13        1.747497e-03        1.701518e+03
```

```
14      1.941525e-03      1.701518e+03
15      6.571252e-03      1.701518e+03
16      1.607637e-03      1.701518e+03
17      1.114232e-03      1.701518e+03
18      9.419803e-04      1.701518e+03
19      9.852313e-04      1.701518e+03
20      1.542826e-05      1.701518e+03
21      1.952793e-09      1.701518e+03
```

In the output, "F(Y)" is the value of the objective function, which in this case, is -2 times the log likelihood function. We notice that it takes 21 iterations till convergence. Suppose we set the maximum number of iterations as 15, then the algorithm stops at the fifteenth iteration and a warning is printed out at the end.

```
Opts.maxIter = 15;
ModelOutput = env(X, Y, u, Opts);
iter      grad              F(Y)
0      1.554710e+02      1.702843e+03
1      1.169893e+01      1.701527e+03
2      1.520058e+01      1.701521e+03
3      2.180808e+00      1.701519e+03
4      3.006970e+00      1.701519e+03
5      9.535630e-01      1.701519e+03
6      1.932848e+00      1.701519e+03
7      6.326229e-01      1.701519e+03
8      5.210354e-01      1.701519e+03
9      5.599613e-01      1.701519e+03
10      3.862568e-01      1.701518e+03
11      2.326454e-03      1.701518e+03
12      1.786087e-03      1.701518e+03
13      1.747497e-03      1.701518e+03
14      1.941525e-03      1.701518e+03
15      6.571252e-03      1.701518e+03
WARNING: reached maximum number of iterations without convergence for
specified tolerances
```

Grassmann manifold optimization can take hundreds of iterations to converge at the default tolerance parameters. We use this example because that we can print its complete iteration process. Now we change the tolerance level of convergence.

```
Opts.ftol = 1e-5;
Opts.gradtol = 1e-3;
ModelOutput = env(X, Y, u, Opts);
iter      grad              F(Y)
0      1.554710e+02      1.702843e+03
1      1.169893e+01      1.701527e+03
2      1.520058e+01      1.701521e+03
```

```
3       2.180664e+00      1.701519e+03
4       3.002431e+00      1.701519e+03
5       9.537034e-01      1.701519e+03
6       1.932020e+00      1.701519e+03
7       6.325922e-01      1.701519e+03
8       5.210506e-01      1.701519e+03
9       5.599522e-01      1.701519e+03
10      4.780483e-01      1.701518e+03
11      3.589887e-03      1.701518e+03
```

As we loosen the tolerance level, it requires less number of iterations till convergence. If the users want to specify their starting value, it can be done by

```
Opts.init = [1 0 0 0 0 0]';
ModelOutput = env(X, Y, u, Opts);
iter      grad              F(Y)
0       1.755092e+02      2.131512e+03
1       5.025313e+02      2.037372e+03
2       1.338489e+02      1.968143e+03
...
59      9.490925e-04      1.701518e+03
60      1.745430e-04      1.701518e+03
61      3.713437e-06      1.701518e+03
```

From the result, we notice that a random starting value normally takes longer to convergence. Furthermore, according to our experience, it is more likely to be trapped in the local minimums, and therefore, random starting values should be avoided.

The Display Argument

In the previous example, the print out for env is the Grassmann manifold optimization process. In this example, we will show different print out types for different functions. We will still use the wheat protein data as the background for the first two cases.

```
load wheatprotein.txt
X = wheatprotein(:, 8);
Y = wheatprotein(:, 1:6);
```

1. With functions on dimension selection, the print out is the current dimension:

```
Opts.verbose = 1;
u = modelselectaic(X, Y, 'env', Opts);
Current dimension 0
Current dimension 1
```

```
Current dimension 2
Current dimension 3
Current dimension 4
Current dimension 5
```

Suppose the number of responses is r, the print out will start with dimension 0 and end with dimension $r-1$. The case of $u = r$ is computed but not printed, because it takes little time, compared to the Grasmann manifold optimization process.

2. With function bootstrapse, the print out is the current number of bootstrap sample.

```
u = 1;
B = 15;
Opts.verbose = 1;
bootse = bootstrapse(X, Y, u, B, 'env', Opts);
Current number of bootstrap sample 1
Current number of bootstrap sample 2
Current number of bootstrap sample 3
Current number of bootstrap sample 4
Current number of bootstrap sample 5
Current number of bootstrap sample 6
Current number of bootstrap sample 7
Current number of bootstrap sample 8
Current number of bootstrap sample 9
Current number of bootstrap sample 10
Current number of bootstrap sample 11
Current number of bootstrap sample 12
Current number of bootstrap sample 13
Current number of bootstrap sample 14
Current number of bootstrap sample 15
```

The print out will start from the first bootstrap sample and end at the last bootstrap sample.

3. With function senv, the print out is the process in the alternating algorithm between the optimization of the scaling parameters and Grassmann manifold.

```
load('sales.txt')
Y = sales(:, 4 : 7);
X = sales(:, 1 : 3);
u = 1;
Opts.verbose = 1;
ModelOutput = senv(X, Y, u, Opts);
Current number of iterations 1
Current number of iterations 2
Current number of iterations 3
Current number of iterations 4
Current number of iterations 5
```

```
Current number of iterations 6
Current number of iterations 7
Current number of iterations 8
Current number of iterations 9
Current number of iterations 10
Current number of iterations 11
Current number of iterations 12
Current number of iterations 13
Current number of iterations 14
Current number of iterations 15
Current number of iterations 16
Current number of iterations 17
Current number of iterations 18
Current number of iterations 19
Current number of iterations 20
Current number of iterations 21
Current number of iterations 22
Current number of iterations 23
Current number of iterations 24
Current number of iterations 25
Current number of iterations 26
```

The maximum number of iterations of the alternating algorithm is 1000. So the print out can run from the first iteration to the thousandth iteration.

4. With other functions, the print out is the Grassmann manifold optimization process as in env.

```
load irisf.mat
d = 1;
Opts.verbose = 1;
ModelOutput = ienv(X, Y, d, Opts)
iter        grad                F(Y)
0       3.064865e+01        2.962639e+03
1       1.679360e+01        2.962025e+03
2       3.560980e+00        2.961025e+03
3       1.431042e-01       2.961019e+03
4       1.667906e-01       2.961019e+03
5       1.699151e-02       2.961019e+03
6       7.836851e-05       2.961019e+03
7       8.623080e-05       2.961019e+03
8       4.470246e-06       2.961019e+03
9       3.716136e-11       2.961019e+03
10       4.629752e-11        2.961019e+03
```

In the display, F(Y) is -2 times the log likelihood function and grad is its derivative.