User Guide for HERMIR version 0.9: Toolbox for Synthesis of Multivariate Stationary Gaussian and non-Gaussian Series

Hannes Helgason, Vladas Pipiras, and Patrice Abry*

June 2, 2011

Contents

1 Organization **3**

2 Introduction **3**
2.1 Component-wise transformations of Gaussian multivariate series 3
2.2 Transformations of independent copies of Gaussian multivariate series 4
2.3 Synthesis problems covered by HERMIR 4

3 Getting started **5**
3.1 Installation instructions 5
3.2 Directory structure and ingredients 6

4 Covariance models implemented in HERMIR **6**
4.1 `CovarExpDecay.m`: Bivariate geometrically decaying covariances 6
4.2 `CovarFARIMA0D0.m`: Multivariate FARIMA$(0, D, 0)$ Series 7
4.3 `CovarMAR1.m`: Multivariate AR(1) Series 8
4.4 `CovarMfGn.m`: Multivariate Fractional Gaussian Noise (FGN) 8

5 Working with non-Gaussian marginals in HERMIR **8**
5.1 Exact covariance mappings 9
5.2 Standard transformation for reaching a prescribed marginal distribution 10

6 Demos **11**
6.1 Multivariate Gaussian synthesis 11
6.2 Univariate non-Gaussian synthesis 12
6.3 Multivariate non-Gaussian synthesis 12
6.4 Monte Carlo simulations 12

*H. Helgason is with KTH, Stockholm (Sweden); V. Pipiras is with UNC, Chapel Hill (USA); P. Abry is with ENS de Lyon (France).

7 Acknowledgements **12**

8 Copyright and warranty **13**

HERMIR is a collection of MATLAB and MEX routines which implements methods for the synthesis of stationary multivariate Gaussian processes. The methodology is described in detail in [2, 3]. HERMIR is freeware and the most current version is available for download at [1]. Feel welcome to send comments and questions to `hannes.helgason@ee.kth.se`.

1 Organization

This is document is organized into the following sections:

2. Introduction – overview of the synthesis problems the toolbox addresses.
3. Getting started – installation instructions and overview of directories.
4. Covariance models – overview of covariance models implemented in toolbox.
5. Non-Gaussian marginals – configuring marginal distributions in HERMIR.
6. Demos – examples demonstrating the toolbox.

2 Introduction

The software's purpose is to provide routines for numerically synthesizing P-variate second-order stationary series

$$Y[n] = (Y_1[n], \ldots, Y_P[n])^T, \quad n = 0, \ldots, N-1, \tag{1}$$

with covariance structure

$$R_Y[n] = EY[0]Y[n]^T - EY[0]EY[n]^T.$$

HERMIR offers two approaches of constructing non-Gaussian series (see Sections 2.1–2.2). See Section 2.3 for two synthesis problems HERMIR can be used for.

2.1 Component-wise transformations of Gaussian multivariate series

The synthesis procedures in HERMIR focus on particular forms of non-Gaussian series which are based on non-linear memoryless transforms

$$Y_p[n] = f_p(X_p[n]), \tag{2}$$

with $X[n] = (X_1[n], \ldots, X_P[n])^T$ a multivariate stationary Gaussian series with mean zero and covariance structure $R_X[n] = EX[0]X[n]^T$.

As explained in [3], the covariance structure $R_X[n]$ of the Gaussian series $X[n]$ and the covariance structure $R_Y[n]$ of the non-Gaussian series $Y[n]$ can be related through the Hermite expansions of f_p, $p = 1, \ldots, P$. The relationship is of the form

$$R_Y[n] = G(R_X[n]), \tag{3}$$

where the mapping $G : \mathbb{R}^{P\times P} \to \mathbb{R}^{P\times P}$ depends on f_p, $p = 1, \ldots, P$ (and is independent of the time index n).

HERMIR offers a way to work numerically with relationships of the form (3) for user-defined functions f_p. The software both allows the user to calculate the mapping $R_Y[n] = G(R_X[n])$, where $R_X[n]$ is given, and provides ways for inverting relationship (3) by numerically calculating

$$R_X[n] = G^{-1}(R_Y[n]); \tag{4}$$

that is, $R_Y[n]$ is given and one wants to find $R_X[n]$ such that (3) hold. This is explained further with MATLAB code examples in Section 5.

2.2 Transformations of independent copies of Gaussian multivariate series

HERMIR also considers the case where the components of $Y[n] = (Y_1[n], \ldots, Y_P[n])^T$ are taken of the form

$$Y_p[n] = f_p(X_p^{(1)}[n], \ldots, X_p^{(K)}[n]), \tag{5}$$

where $X^{(k)}[n] = (X_1^{(k)}[n], \ldots, X_P^{(k)}[n])^T$, $k = 1, \ldots, K$, are i.i.d. copies of a stationary Gaussian series $X[n]$ each with the same covariance structure $R_X[n]$. This type of construction is of interest since it offers synthesis of many marginals of practical use using simple formulas with nice analytic properties.

Here again one seeks a way to relate the Gaussian covariance structure $R_X[n]$ with the non-Gaussian covariance structure $R_Y[n]$. As discussed in [3], one can here also relate the covariances using an expression of the form (3) (this time using *multivariate* Hermite expansions).

In many practical cases the covariances can be related using simple analytic expressions, allowing one to easily evaluate both the *forward mapping* $R_Y[n] = G(R_X[n])$ and the *backward mapping* $R_X[n] = G^{-1}(R_Y[n])$. In HERMIR one can find a list of constructions for popular marginals having such "exact" covariance mappings; the list of marginals along with MATLAB code examples are given in Section 5. (See [3] for details about these constructions.)

2.3 Synthesis problems covered by HERMIR

HERMIR can be used in the following synthesis problems:

- **Synthesis of multivariate Gaussian series:** HERMIR offers a fast way, using circulant matrix embedding techniques and FFT algorithms, to synthesize stationary multivariate Gaussian series (i.e., the case when $f_p(x) = x$).
- **Forward Problem:** Prescribe transforms f_p, $p = 1, \ldots, P$, and covariance structure $R_X[n]$ for underlying Gaussian series $X[n]$. Synthesize $Y[n]$ and calculate the resulting covariance $R_Y[n]$.
- **Backward Problem:** Prescribe targeted covariance structure $R_Y[n]$ and transforms f_p, $p = 1, \ldots, P$. Find $R_X[n]$ such that $Y[n]$ has covariance $R_Y[n]$.

Remark for Backward Problem: Only prescribing marginal distributions and a covariance structure will, in general, not uniquely define the series Y; that is, more than one series can have the same prescribed marginal distributions and second-order moments. Another important point to make is that not all joint choices of marginal distributions and covariance structures are possible. See [3] for detailed discussion about these two issues.

In cases where prescribed covariance structure $R_Y[n]$ cannot be reached, HERMIR can be used to find a covariance structure which approximates the targeted covariance in a certain optimal fashion (see [3]). This approximating covariance can then be used for synthesis.

3 Getting started

3.1 Installation instructions

1. *Requirements:* The software requirements for HERMIR are

   - `tar` and `gunzip` to install the package on Unix-based systems, or `zip` to install the package on Windows.
   - MATLAB – the code has been tested on MATLAB version 7.10 but should also work on some older versions.

2. HERMIR is distributed as a compressed tar or zip file. To install,

   (a) Download the compressed archive.

   (b) Uncompress it at the desired location:

   ```
   gunzip -c Hermir0_9.tar.gz | tar xfv -
   ```

   or use zip if installing under Windows.

   This will create a directory tree rooted at `Hermir0_9` containing the source code.

3. Before using HERMIR, the MATLAB paths must be set up. Here we explain two ways:

   (a) **Starting HERMIR on demand:**
   Each time after starting MATLAB, run `HermirPath.m` from the MATLAB command prompt. You have to be located in the directory HERMIR directory for this to work unless you edit the file `HermirPath.m` and change the variable `MYHERMIR` to the full path of the directory where HERMIR is installed.

   (b) **Permanently adding HERMIR to MATLAB path:**
   To permanently add the HERMIR directories to your `MATLABPATH` you can edit the file `startup.m`; this file (if it exists), should be located in the default or current startup folder, which is where MATLAB first looks for it. For more information regarding `startup.m` consult MATLAB's help documentation.

   Once you have located or created a `startup.m` file, follow the following steps:

   i. Edit the file `HermirPath.m` and change the variable `MYHERMIR` to the directory where HERMIR is installed

ii. Copy `HermirPath.m` to the same folder as the file 'startup.m'

iii. Add a line with the command '`HermirPath`' to the file `startup.m`

4. To test that HERMIR is installed properly, start MATLAB, and run the demo script

```
DemoCovarMAR1
```

If HERMIR directories are not permanently added to your `MATLABPATH`, go into the HERMIR root directory in your MATLAB command prompt and set up the paths by executing

```
HermirPath
```

3.2 Directory structure and ingredients

The routines in HERMIR are organized into several folders. Here is a list of the main folders with rough descriptions of their ingredients:

`GaussGen`	– routines for the generation of stationary multivariate Gaussian series.
`NonGaussGen`	– routines for the generation of stationary multivariate non-Gaussian series.
`CovarGen`	– routines for generating covariance structures (see Section 4).
`Demo`	– sample scripts demonstrating the code (see Section 6).
`Documentation`	– documentation for HERMIR, along with copyright and warranty notices.
`Utilities`	– various utility routines.

4 Covariance models implemented in HERMIR

The folder `CovarGen` stores the following routines for generating the several different covariance structures:

`CovarExpDecay.m`	– Bivariate geometrically decaying covariance.
`CovarFARIMA0D0.m`	– Multivariate FARIMA$(0, D, 0)$ covariance.
`CovarMAR1.m`	– Multivariate AR(1) covariance.
`CovarMfGn.m`	– Multivariate fractional Gaussian noise covariance.

These covariances were given as examples and discussed in [2]. Short descriptions are given below; see also the documentation given in the MATLAB functions, for example, by using the '`help`' command from the MATLAB command prompt (e.g., run '`help CovarMAR1`' to get documentation for `CovarMAR1.m`).

4.1 `CovarExpDecay.m`: Bivariate geometrically decaying covariances

This function generates the covariance structure given by

$$R[n] = \begin{pmatrix} \varphi_1^n & c\varphi_3^n \\ c\varphi_3^n & \varphi_2^n \end{pmatrix},$$

where $n \geq 0$, $0 < \varphi_1, \varphi_2 < 1$ and $c, \varphi_3 \in \mathbb{R}$. As explained in [2], synthesis using the circulant matrix embedding method described therein and implemented in HERMIR should work at least when the parameters satisfy the conditions

$$0 \leq c \leq 1, \qquad |\varphi_3| \leq \min(\varphi_1, \varphi_2),$$

and

$$1 - \frac{1 - \max(\varphi_1, \varphi_2)}{\sqrt{c}} \leq \varphi_3 \leq \min(\varphi_1, \varphi_2).$$

Parameters: The covariance structure in `CovarMAR1.m` is configured by φ_1, φ_2, φ_3, and c.

MATLAB example: See `DemoCovarExpDecay.m` in the `Demo` folder.

4.2 `CovarFARIMA0D0.m`: Multivariate FARIMA$(0, D, 0)$ Series

This function generates the covariance structure for FARIMA$(0, D, 0)$ series $X[n]$ defined by

$$(I - B)^D X[n] = \epsilon[n],$$

where B is the backshift operator. The innovations $\epsilon[n]$ are i.i.d. Gaussian vectors of length P with mean zero and covariance matrix Σ_ϵ. Assuming that D is diagonalizable as $D = W\Lambda W^{-1}$, with diagonal $\Lambda = diag(d_1, \ldots, d_P)$, $|d_k| < 1/2$, one can write the series as

$$X[n] = W X_\Lambda[n],$$

where $X_\Lambda[n]$ is a FARIMA$(0, \Lambda, 0)$ series with the innovations $\eta[n]$ having the covariance

$$\Sigma_\eta = (\Sigma_{\eta,p,p'})_{1 \leq p,p' \leq P} = W^{-1} \Sigma_\epsilon (W^{-1})^*,$$

where A^* stands for the conjugate-transpose of a matrix A. The covariance of $X[n]$ is given by

$$R[n] = W R_\Lambda[n] W^*,$$

where $n \geq 0$ and the entries of $R_\Lambda[n]$, the covariance of $X_\Lambda[n]$, are

$$R_\Lambda[n]_{p,p'} = \Sigma_{\eta,p,p'} \frac{(-1)^n \Gamma(1 - d_p - \overline{d}_{p'})}{\Gamma(1 + n - \overline{d}_{p'}) \Gamma(1 - n - d_p)},$$

where $\Gamma(\cdot)$ stands for the gamma-function and $\overline{z}$ stands for the complex-conjugate of a complex number z. (For further details about this covariance model, see [2] and references therein.)

Parameters: The parameters configuring the covariance structure generated by `CovarFARIMA0D0.m` correspond to the three parameters W, $\Lambda = diag(d_1, \ldots, d_P)$, and Σ_ϵ above. Recall that $D = W\Lambda W^{-1}$ and Σ_ϵ is the covariance matrix for the innovations of the FARIMA$(0, D, 0)$ series.

MATLAB example: See `DemoCovarFARIMA0D0.m` in the `Demo` folder.

4.3 `CovarMAR1.m`: Multivariate AR(1) Series

This function generates covariance structure for multivariate series of the form

$$X[n] = \Phi X[n-1] + \epsilon[n],\ n \in \mathbb{Z},$$

where Φ is a $P \times P$ matrix whose eigenvalues are smaller than 1 in absolute value, and $\epsilon[n]$ are i.i.d. Gaussian vectors of length P with mean zero and covariance $E\epsilon[n]\epsilon[n]^T = \Sigma_\epsilon$.

Parameters: The parameters configuring the covariance structure generated by `CovarMAR1.m` are Φ and Σ_ϵ.

MATLAB example: See `DemoCovarMAR1.m` in the `Demo` folder.

4.4 `CovarMfGn.m`: Multivariate Fractional Gaussian Noise (FGN)

This function generates the following multivariate FGN covariance structure

$$R[n] = \frac{1}{2}\left(|n+1|^H \Sigma |n+1|^{H^*} + |n-1|^H \Sigma |n-1|^{H^*} - 2|n|^H \Sigma |n|^{H^*}\right),$$

with $H = W\Lambda W^{-1}$, with diagonal $\Lambda = \mathrm{diag}(h_1, \ldots, h_P)$, $0 < h_k < 1$. The matrix Σ is given by

$$\Sigma = 8W\Sigma_0 W^*,$$

with the entries $\Sigma_{0,p,p'}$ of Σ_0 given by

$$\Sigma_{0,p,p'} = -\frac{c_{p,p'}}{2}\Gamma(-h_p - \overline{h}_{p'}) \cos\left(\frac{(h_p + \overline{h}_{p'})\pi}{2}\right),$$

where $C = (c_{p,p'})_{1 \le p,p' \le P} = W^{-1}AA^*(W^{-1})^*$, with A a real-valued $P \times P$ matrix.

Parameters: The parameters configuring the covariance structure generated by `CovarMfGn.m` correspond to the three parameters W, Λ, and A above.

MATLAB example: See `DemoCovarMfGn.m` in the `Demo` folder.

5 Working with non-Gaussian marginals in HERMIR

Here we give details about how to configure marginal distributions in HERMIR based on the two ways of synthesizing non-Gaussian series $Y[n]$ discussed in Sections 2.1–2.2. The folder `NonGaussGen` stores MATLAB routines for working with non-Gaussian marginals; some of the most relevant are:

- `GenUnivarNonGaussUsingExactMap.m`:
  For generating *univariate* stationary non-Gaussian series for a list of cases where one has analytic formulas for covariance mappings. See Section 5.1.

- `GenMVnGaussUsingExactMap.m`:
  For generating *multivariate* stationary non-Gaussian series for a list of cases where one has analytic formulas for covariance mappings. See Section 5.1.

- `MapAutoCovar.m`, `MapCovar.m`:
  Functions for calculating mapping between non-Gaussian and Gaussian covariances for the list of transformations given in Section 5.1. `MapAutoCovar.m` is used by `GenUnivarNonGaussUsingExactMap.m` and `MapCovar.m` is used by `GenMVnGaussUsingExactMap.m`.

- `GenUnivarNonGaussUsingHermite.m`:
  For generating *univariate* stationary non-Gaussian series using Hermite expansions.

- `GenMVnGaussUsingHermite`:
  For generating *multivariate* stationary non-Gaussian series using Hermite expansions.

5.1 Exact covariance mappings

Here we list a collection of marginals which can be easily reached using construction (5) in Section 2.2. We consider here two families of transforms for reaching important marginals. One transform is of the type

$$f_p(x_1, \ldots, x_K) = \sum_{k=1}^{K} b_k x_k^2,$$

where (b_k) is a collection of deterministic scalars; see Table 1 for a list of popular marginals that can be reached this way.

Table 1: Important marginals resulting from $f_p(x) = \sum_{k=1}^{K} b_k x_k^2$.

Marginal Type	Parameter Values
Exponential distribution with mean $a > 0$	$K = 2$, $b_1 = b_2 = a/2$
Laplace$(0, a)$ distribution with zero mean and variance $2a^2$	$K = 4$, $b_1 = b_3 = a/2$, $b_2 = b_4 = -a/2$
Chi-square distribution with ν degrees of freedom	$K = \nu$, $b_k = 1, \forall k$
Erlang(α, β) distribution, i.e., sum of α exponential variables, each with mean $\beta > 0$	$K = 2\alpha$, $b_k = \beta/2, \forall k$

The other family contains transforms of the type

$$f_p(x_1, \ldots, x_K) = \exp\left(\sum_{k=1}^{K} b_k x_k^2\right),$$

where (b_k) is a collection of deterministic scalars; see Table 2 for a list of popular marginals that can be reached this way.

Table 2: Important marginals resulting from $f_p(x) = \exp(\sum_{k=1}^{K} b_k x_k^2)$.

Marginal Type	Parameter Values
Pareto distibution with minimum 1 and index $\alpha > 0$	$K = 2$, $b_1 = b_2 = \frac{1}{2\alpha}$
Uniform$(0, 1)$ distibution	$K = 2$, $b_1 = b_2 = -1/2$

If the marginals of all the components Y_p, $p = 1, \ldots, P$, are the same, that is,

$$f_1 = \cdots = f_P,$$

the transforms in Tables 1–2 allow one to have simple analytic expressions for calculating both the forward covariance mapping (3)

$$R_Y[n] = G(R_X[n]),$$

and the backward covariance mapping (4)

$$R_X[n] = G^{-1}(R_Y[n]).$$

MATLAB code: The most relevant MATLAB functions in HERMIR related to the transformations in Tables 1–2 are located in the folder `NonGaussGen`. These are the functions:

- `GenUnivarNonGaussUsingExactMap.m`:
  For generating *univariate* stationary non-Gaussian series.
- `GenMVnGaussUsingExactMap.m`:
  For generating *multivariate* stationary non-Gaussian series.
- `MapAutoCovar.m`, `MapCovar.m`:
  Function for calculating mapping between non-Gaussian and Gaussian covariances.

For a demonstration, see `DemoGenUnivarNonGauss.m` located in the `Demo` folder.

5.2 Standard transformation for reaching a prescribed marginal distribution

Consider the case where for the pth coordinate in the series (1) one targets a marginal with a cdf F_p; that is, for each n,

$$Y_p[n] \sim F_p.$$

With the construction $Y_p[n] = f_p(X_p[n])$ from (2) used in HERMIR, there are infinitely many possibilities of transformations f_p which can achieve this (this is discussed in [3]). One natural choice for targeting the cdf F_p is

$$f_p(x) = F_p^{-1}(\Phi(x)), \qquad \text{where } \Phi \text{ is the cdf for } \mathcal{N}(0, 1). \tag{6}$$

In many cases MATLAB provides a very compact way of defining these type of transformations using *function handles* (see MATLAB documentation). For example, if one is targeting a χ^2_1 marginal, the following MATLAB code lines will return a function handle in the variable `funch`:

```
v = 1; % number of degrees of freedom
funch = @(x) chi2inv(normcdf(x),v);
```

Then one can simply use the variable `funch` as any other MATLAB function; here is an example from the MATLAB command prompt:

```
>> funch(1)
ans =
    1.9870
```

To define function handles for general χ^2_ν distributions, the variable `v` above has just to be set to the number of degrees of freedom ν. This compact way of using inline function handles for defining transformations of the type (6) in MATLAB, can be used if MATLAB function for the inverse cdf F_p^{-1} is available.

6 Demos

6.1 Multivariate Gaussian synthesis

The functions most relevant for synthesis of multivariate Gaussian series are in the folder `GaussGen`. For the synthesis of bivariate stationary Gaussian series, one can use the function `GenBivarGauss.m` as in the following MATLAB example, built on file `DemoCovarMAR1.m`:

```
% Set requested length of synthesized series
N = 2^10+1;

% Generate MAR(1) covariance structure
Phi = [0.8 1;0 0.2]; Sigmae = eye(2);
R = CovarMAR1(N,Phi,Sigmae);

% Generate bivariate Gaussian series
[X1,X2] = GenBivarGauss(R,N);
```

This example will synthesize bivariate series $X[n] = (X_1[n], X_2[n])^T$, $n = 0, \ldots, N-1$, with multivariate AR(1) covariance structure configured by the parameters `Phi` and `Sigmae`. The variable `X1` stores a realization of the component $X_1[n]$ and `X2` stores a realization of the component $X_2[n]$; both variables are vectors of length N.

An alternative (and slower) way of doing this is by using the more general function `GenMultivarGauss.m` which can be used when the number of components P is greater than or equal to 2. In the above example, this function can be used by replacing '`[X1,X2] = GenBivarGauss(R,N)`' by

```
X = GenMultivarGauss(R,N);
```

The variable `X` stores a realization of the component $X_1[n]$ in the row vector `X(1,:)` and a realization of the component $X_2[n]$ in the row vector `X(2,:)`.

6.2 Univariate non-Gaussian synthesis

For a demonstration for synthesis univariate non-Gaussian series based on the transformations given in Section 5, see `DemoGenUnivarNonGauss.m` in the `Demo` folder.

Documented demonstrations related to synthesis of univariate non-Gaussian series based on Hermite expansions are given in `DemoGenUnivarNonGauss.m`, `DemoGenUnivarHermiteDetailed.m`, and `DemoCovarSequence.m`.

6.3 Multivariate non-Gaussian synthesis

For a demonstration for synthesis univariate non-Gaussian series based on the transformations given in Section 5, see `DemoGenMVnGcovmap.m` in the `Demo` folder.

For documented demonstrations related to synthesis of multivariate non-Gaussian series based on Hermite expansions, see `DemoMonteCarloHermite.m`.

6.4 Monte Carlo simulations

In Monte Carlo simulations, one only needs to do the initialization step in the synthesis procedures once for every choice of prescribed covariance and marginal. This initialization step involves inversion of the covariance relationship (i.e., solve (4) numerically) and a call to the function `InitStepMultivarGaussBestApprox.m` or `InitMVnGaussUsingHermite.m`. In the `Demo` folder one can find documented demonstrations showing how one can decrease the cost of generating realizations for Monte Carlo simulations by doing this initialization only once. See the following MATLAB scripts:

- `DemoMonteCarloHermite1D.m`:
  Monte Carlo simulations using *multivariate* stationary non-Gaussian series based on Hermite expansion construction.
- `DemoMonteCarloHermite.m`:
  Monte Carlo simulations using *multivariate* stationary non-Gaussian series based on Hermite expansion construction.
- `DemoMonteCarloExactMap1D.m`:
  Monte Carlo simulation using univariate non-Gaussian series based on exact covariance mapping.
- `DemoMonteCarloExactMap.m`:
  Monte Carlo simulation using multivariate non-Gaussian series based on exact covariance mapping.

7 Acknowledgements

Contributors to Hermir are Hannes Helgason (KTH, Stockholm, Sweden), Vladas Pipiras (UNC, Chapel Hill, USA), and Patrice Abry (ENS de Lyon, France).

8 Copyright and warranty

HERMIR is copyrighted material. There is no warranty attached to HERMIR. For copying permissions and warranty notice, see the documents `COPYING.txt` and `WARRANTY.txt` located in the folder `Documentation` in the HERMIR distribution.

References

[1] HERMIR. Website: `http://www.hermir.org`.

[2] H. Helgason, V. Pipiras, and P. Abry. Fast and exact synthesis of stationary multivariate Gaussian time series using circulant embedding. *Signal Processing*, 91(5):1123–1133, 2011.

[3] H. Helgason, V. Pipiras, and P. Abry. Synthesis of multivariate stationary series with prescribed marginal distributions and covariance using circulant matrix embedding. *Signal Processing*, 91(8):1741–1758, 2011.