

Strohe, Hans Gerhard; Härdle, Wolfgang; Geppert, Frank

**Working Paper**

## DPLS in XploRe: A PLS approach to dynamic path models

SFB 373 Discussion Paper, No. 1999,80

**Provided in Cooperation with:**

Collaborative Research Center 373: Quantification and Simulation of Economic Processes,  
Humboldt University Berlin

*Suggested Citation:* Strohe, Hans Gerhard; Härdle, Wolfgang; Geppert, Frank (1999) : DPLS in XploRe: A PLS approach to dynamic path models, SFB 373 Discussion Paper, No. 1999,80, Humboldt University of Berlin, Interdisciplinary Research Project 373: Quantification and Simulation of Economic Processes, Berlin,  
<https://nbn-resolving.de/urn:nbn:de:kobv:11-10047007>

This Version is available at:

<https://hdl.handle.net/10419/61763>

**Standard-Nutzungsbedingungen:**

Die Dokumente auf EconStor dürfen zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden.

Sie dürfen die Dokumente nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, öffentlich zugänglich machen, vertreiben oder anderweitig nutzen.

Sofern die Verfasser die Dokumente unter Open-Content-Lizenzen (insbesondere CC-Lizenzen) zur Verfügung gestellt haben sollten, gelten abweichend von diesen Nutzungsbedingungen die in der dort genannten Lizenz gewährten Nutzungsrechte.

**Terms of use:**

*Documents in EconStor may be saved and copied for your personal and scholarly purposes.*

*You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.*

*If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.*

University of Potsdam and Humboldt University Berlin

*Hans Gerhard Strohe, Wolfgang Haerdle, Frank Geppert*

**DPLS IN XPLORE**  
**A PLS APPROACH TO DYNAMIC PATH MODELS**

Paper to be presented at the  
International Symposium on PLS Methods  
Jouy-en-Josas, France  
October 5-6, 1999

*Hans Gerhard Strohe, Potsdam; Wolfgang Haerdle, Berlin Frank Geppert, Potsdam*

# **DPLS IN XPLORE**

## **A PLS APPROACH TO DYNAMIC PATH MODELS**

### **1 INTRODUCTION**

In the presentation, a computational approach to dynamic modelling with latent variables will be shown. It has been developed on the base of H. Wold's Partial Least Squares (PLS). An operator matrix containing the lag operator  $L$  is substituted for the path coefficient matrix of Wold's static PLS model. On what is called the dynamic PLS model (DPLS) the original PLS estimation algorithm is virtually applicable. The characteristic of dynamic modelling is the usage of time series and lags, i.e. autoregressive terms and distributed lags. DPLS combines both of them. Lagged and leaded latent variables are used in the iterative process of estimating the weights of the manifest variables. The path coefficients are estimated by OLS or an appropriate dynamic modelling method. The redundancy coefficient allows measuring the forecasting validity. DPLS has been programmed in the statistical computing environment XploRe. Some properties of DPLS can be shown by Monte Carlo simulation. Finally DPLS model with latent financial market and national economy variables of Germany is to be presented.

### **2 THE PRINCIPLES OF DPLS**

#### **2.1 Partial Least Squares by H. Wold**

The Partial Least Squares algorithm (PLS) was developed by Herman Wold (1974) as an explorative approach to estimating path models.

The inner PLS model can be written in a structural form

$$\eta_t = \mathbf{B}\eta_t + v_t \quad t = 1, \dots, T \quad (1)$$

where  $\eta_t$  is a vector of  $K$  latent variables with zero mean and unit variance and  $v_t$  a disturbance term with zero expectation. The  $K \times K$ -matrix  $\mathbf{B}$  is considered to be triangular

i.e. the model is expected to be recursive.  $\eta_t$  can be measured by the manifest variable vector  $\mathbf{y}_t$  (outer PLS model):

$$\mathbf{y}_t = \mathbf{P}\eta_t + \varepsilon_t \quad \text{and} \quad \eta_t = \mathbf{W}'\mathbf{y}_t \quad (2)$$

where  $\mathbf{P}$  is a  $M \times K$  matrix of loadings and  $\mathbf{W}$  a  $M \times K$  matrix of weights. The matrices  $\mathbf{P}$  and  $\mathbf{W}$  can be made block diagonal corresponding to the blocks of manifest variables belonging to the individual latent variables. The error term  $\varepsilon_t$  is white noise with zero mean.

In this form, the model was not able to cover dynamics, i.e. lagged relationships, without introducing the lagged variables as additional variables with weights and loadings different from those of the unlagged variables.

## 2.2 Dynamic Partial Least Squares

If a path model contains for at least one latent variable a lagged relationship besides the normal contemporary dependencies it is referred to as dynamic path model. If the model parameters are estimated by PLS it is called a dynamic partial least squares or DPLS model.

In order to estimate a Dynamic Partial Least Squares model firstly a path model for the relationships between manifest and latent variables should be designed. In addition to the "usual" i.e. unlagged relationships, there will be special paths for "lagged" relationships i.e. dependencies on the lagged latent variables without introducing these lagged variables as individual independent variables. That makes the difference to traditional PLS path models.

The inner model (1) can be made dynamic by implementing a first order lagged term:

$$\eta_t = \mathbf{B}\eta_t + \mathbf{C}\eta_{t-1} + v_t \quad (3)$$

where  $\mathbf{C}$  is a  $K \times K$  matrix of vector autoregression coefficients. In order to complete the PLS path model, the weight relation

$$\eta_t = \mathbf{W}'\mathbf{y}_t \quad (4)$$

is to be added to model (3).

The dynamic form of the structural model can be transformed into the exterior shape of the "normal" PLS model:

$$\eta_t = \mathbf{F}\eta_t + v_t \quad (5)$$

where

$$\mathbf{F} = \mathbf{B} + \mathbf{C}\mathbf{L} \quad (6)$$

is an operator matrix containing the lag operator  $\mathbf{L}$  defined by  $\mathbf{L}x_t = x_{t-1}$ .

On the now dynamic PLS model (5), (4), the original PLS algorithm is applicable in formal analogy (DPLS):

Initially, Boolean design matrices  $\mathbf{D}_B$ ,  $\mathbf{D}_C$  and  $\mathbf{D}_P$  corresponding to the unlagged and lagged dependencies in the inner model (5) and to the outer model (4) must be fixed. The inner design matrix  $\mathbf{D}_B$  contains the figure „1“ if there is a connection between two latent variables in the path model and consists of zeros elsewhere. Similarly, the lag design matrix  $\mathbf{D}_C$  consists of ones and zeros corresponding to whether or not there is assumed to be first order lagged (auto-)regression between latent variables.  $\mathbf{D}_P = [d_{mk}]$  is the outer design matrix consisting of 1 or 0 corresponding to whether or not a variable  $y^m$  of  $\mathbf{Y}$  belongs to the block of a certain latent variable i.e. a row  $\eta^k$  of  $\mathbf{H}$ .

In order to estimate the weight matrix  $\mathbf{W}$  firstly an initial representation of the latent variables as components of the manifest variables with chosen starting values for the matrix  $\mathbf{W}$  is to be calculated:  $\eta_t = \mathbf{W}'y_t$ . The initial latent variables are standardised to unit variance. Then what is called "neighbourhood" variables are calculated corresponding to the inner path model and taking into account each relationship for both the dependent and the independent variables:

$$\begin{aligned}\eta_t^* &= \mathbf{F}^* \eta_t \\ &= (\mathbf{D}_B + \mathbf{D}_B') * \mathbf{R} \eta_t + \mathbf{D}_C * \mathbf{A} \eta_{t-1} + (\mathbf{D}_C * \mathbf{A})' \eta_{t+1}\end{aligned}\quad (9)$$

The matrices

$$\mathbf{R} = \mathbf{H}\mathbf{H}'/T \quad \text{and} \quad \mathbf{A} = \mathbf{H}(\mathbf{L}\mathbf{H})'/T$$

are the correlation matrix and an approximation for the first order autocorrelation matrix of the latent variables, respectively, with  $\mathbf{H} = (\eta_1, \eta_2, \dots, \eta_T)$  being the matrix of the latent variable vectors and  $\mathbf{L}\mathbf{H} = (\eta_0, \eta_1, \dots, \eta_{T-1})$ . The symbol \* denotes element wise multiplication.

New values of the weight matrix  $\mathbf{W} = (\omega_{mk})$  are gained by OLS estimation of the parameters of the regression equation

$$y_t^m = \omega_{mk} \eta_t^{k*} + v_t^{mk}$$

Using this new weight matrix we repeat the procedure. The iteration process is considered to converge when subsequent estimations of the latent variables scores  $\eta_t$  do not relevantly differ from the previous ones.

Then the coefficient matrices  $\mathbf{B}$  and  $\mathbf{C}$  of the inner model (5)

$$\eta_t = (\mathbf{B} + \mathbf{C}\mathbf{L}) \eta_t + v_t$$

can be estimated by a suitable method for dynamic models, depending on the dynamic properties of the latent variables, e.g. OLS or GLS.

The loadings  $\mathbf{P}$  of the outer model

$$y_t = \mathbf{P} \eta_t + \varepsilon_t$$

are estimated by simple OLS. More details of the whole procedures can be found in Strohe (1996, 1997)

In order to obtain goodness-of-fit measures we calculate the ratio of two diagonal variance matrices:

$$\mathbf{G} = (\mathbf{I} * \mathbf{G}^*) (\mathbf{I} * \Sigma_y)^{-1} \quad (10)$$

$$\text{where } \mathbf{G}^* = \mathbf{PBRB}'\mathbf{P}' + \mathbf{PCRC}'\mathbf{P}' + \mathbf{PBAC}'\mathbf{P}' + (\mathbf{PBAC}'\mathbf{P}')' \approx \text{cov}(\mathbf{Y}^*) \quad (11)$$

(see Strohe 1997) and  $\Sigma_y$  denotes the empirical covariance matrix of the manifest variables. The figures in the diagonal of  $\mathbf{G}$ , expressing to what extent the empirical variance of each manifest variable is reproduced by the model, are called redundancies of the individual variables.

The average of these measures

$$G^2 = \text{trace } \mathbf{G} / M \quad (12)$$

is the redundancy coefficient or average redundancy and is used for the evaluation of the goodness of fit of the model as a whole.

### 2.3 The computer programme DPLS

The very sophisticated computer program LVPLS for static partial least squares models was developed by Lohmoeller (1984, 1989). Unfortunately, it is not effectively applicable to dynamic path models.

The computer programme DPLS (Strohe, Geppert 1997) is available as both PC-ISP (1992) and XploRe (Haerdle et.al. 1995) macros. The syntax to control the programme is nearly identical in both versions. It consists of three basic modules. The first one puts the DPLS algorithm within the XploRe environment into action, the second one calculates a redundancy measure and the last one is a tool for easier creating input-variables. This one works on the base of several menus and dialogs. So it is not necessary to know all input matrices exactly in advance and the only input to be prepared by the use of ordinary XploRe commands is the indicator matrix.

The further input matrices firstly are to be created and defined by the above mentioned tool (third basic module) or 'by hand'

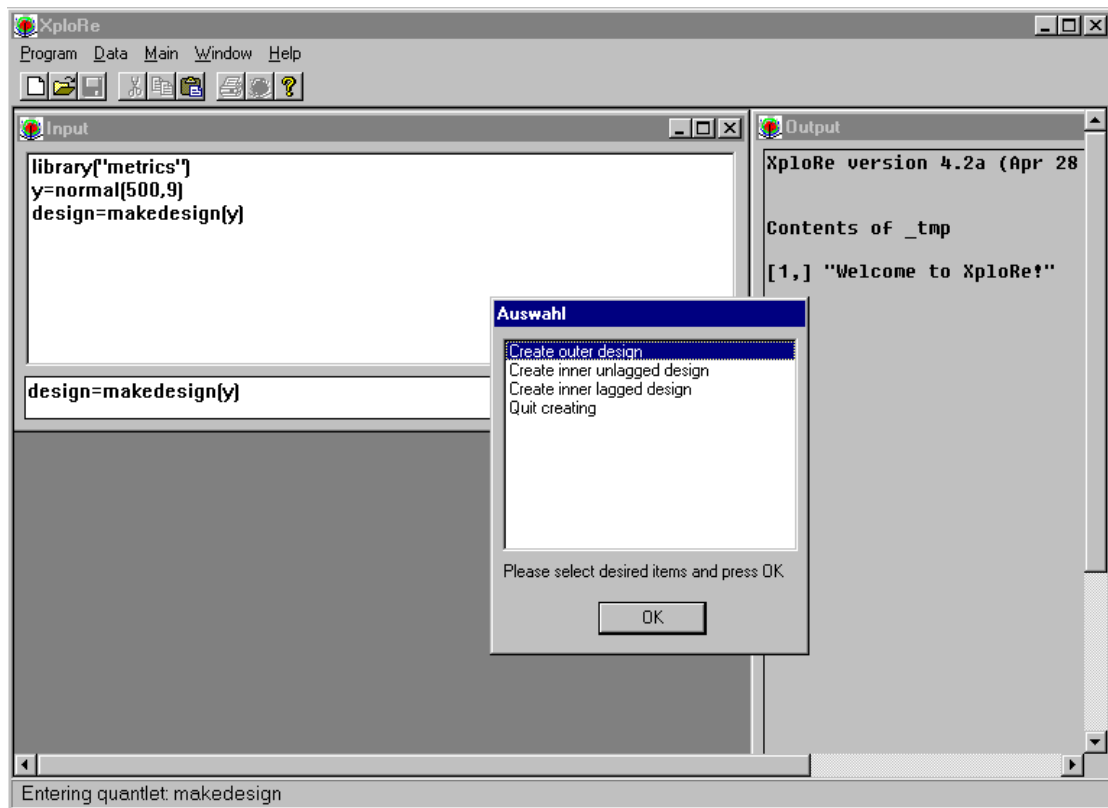
But if an user wants to calculate a model several times one after the other and if he/she wants to modify parameters on the fly in a programmed simulation or something like this he/she can take an advantage by using the basic programmes directly. In that case one has to know more about the details of input and output of the corresponding module.

At first, for running a DPLS session, XploRe has to be started. One must load the library into memory by using the "library" command. XploRe users have to type:

```
Library("metrics") # loads commands and functions
```

First, a design-making session (figure 1) could be started by:

```
design=makedesign(y)
```



**Figure 1:** A session for creating design-matrices in XploRe

There  $y$  is a  $n \times l$ -matrix with manifest variables (indicators). The output  $design$  contains several variables. Among them the  $dy$ -matrix ( $l \times k$ ) with outer designs (0 or 1). Rows are counting manifest variables. Furthermore,  $d$  is a  $k \times k$  matrix with inner unlagged designs (0 or 1). No diagonal values are allowed. The  $dl$ -matrix ( $k \times k$ ) contains the inner lagged designs (0 or 1). Diagonal elements are showing autoregression. The  $w$ -matrix represents start weights with same dimensions as  $dy$ . It is simply a copy of  $dy$ .

It is possible to address these variables in a container-variable with the point as an separator between them both. In the case above:

```
design.dy
design.w
design.dl
design.w
```

The syntax of *makedesign* is to be seen below:

*Usage:* Out = makedesign(y)

*Input:*

y a nxl-matrix of manifest variables (indicators)

*Output:*

Out.dy a lxx matrix of outer designs (0 or 1) rows are counting manifest variables  
 Out.d a kxx matrix of inner unlagged designs (0 or 1) no diagonal values allowed  
 Out.dl a kxx matrix of inner lagged designs (0 or 1) diagonal elements are showing autoregression  
 Out.w a matrix of start weights same dimensions as dy will be a copy of dy

Besides that, it is possible to create all matrices by hand. An easy sample could be seen below:

```

library("metrics")
randomize(13409)
b1=0.3
c1=0.6
s=500
n1=normal(s+1)
n1lag=n1[1:s,]
n1=n1[2:rows(n1),]
;inner model
n2=b1*n1+c1*n1lag+normal(rows(n1))/5
n=n1 ~ n2

nn=n./sqrt(var(n))
;loadings matrix
p=(1 | 2 | 3 | 4 | 0 | 0 | 0)~(0 | 0 | 0 | 0 | 5 | 6 | 7)
y=nn*p'+normal(rows(n),rows(p))/8
d=(0 | 1) ~ (0 | 0)
dl=(0 | 1) ~ (0 | 0)
w=(1 | 1 | 1 | 1 | 0 | 0 | 0)~(0 | 0 | 0 | 0 | 1 | 1 | 1)
myfit=dpls(w,d,w,dl,y,3)
myfit.b
myfit.sk
myfit.sk1

```

As one could see above, a common session of DPLS (figure 2) can be started by:

*myfit=dpls(w,d,dy,dl,y,acc)*

There *dpls* denotes the programme used and *w* is a term for the weights the algorithm is to start with. The term *d* means the inner and *dy* the outer design matrix. Furthermore, *dl* is a symbol for an inner design matrix as well but it contains the lagged connections in the model. The matrices *d*, *dy* *dl* represent our idea about the model structure. The matrix *y* contains the time series of all indicators (manifest variables) and represents virtually all empirical information available. The digit *acc* stands for “accuracy” and controls the final stop of the iteration process. DPLS uses this number to check after every iteration whether or not the new calculated values are significantly different from the previously calculated values *acc* specifying how many decimals are taken into consideration.

The syntax of *dpls* has the following structure:



*Usage:* `myfit = dpls(w,d,dy,d1,y,genau)`

*Input:*

`w` a matrix of start weights same dimensions as `dy`  
`d` a `kxk` matrix of inner unlagged designs (0 or 1) no diagonal values allowed  
`dy` a `lxk` matrix of outer designs (0 or 1) rows are counting manifest variables  
`d1` a `kxk` matrix of inner lagged designs (0 or 1) diagonal elements are showing autoregression  
`y` a `nxl` matrix of manifest variables (indicators)  
`genau` a scalar of canceling criterion

*Output:*

`myfit.wg` a matrix of weights  
`myfit.b` a matrix of loadings  
`myfit.sk` a matrix of path coefficients with dimensions like `d` (`kxk`)  
`myfit.sk1` a matrix of lagged path coefficients with dimensions like `d` (`kxk`) and ordered like designed  
`myfit.lk` a matrix of latent variables  
`myfit.iter` number of iterations

```

XploRe
Program Data Main Window Help
Editor: - noname.xpl
library("metrics")
randomize(13409)
b1=0.3
c1=0.6
s=500
n1=normal(s+1)
n1lag=n1[1:s,]
n1=n1[2:rows(n1),]
;inner model
n2=b1*n1+c1*n1lag+normal(rows(n1))/5
n=n1 ~ n2
nn=n./sqrt(var(n))
;loadings matrix
p=( 1 | 2 | 3 | 4 | 0 | 0 | 0) ~ (0 | 0 | 0 | 0 | 5 | 6)
y=nn*p'+normal(rows(n),rows(p))/8
d=(0 | 1) ~ (0 | 0)
d1=(0 | 1) ~ (0 | 0)
w=(1 | 1 | 1 | 1 | 0 | 0 | 0) ~ (0 | 0 | 0 | 0 | 1 | 1)
myfit=dpls(w,d,w,d1,y,3)
myfit.b
myfit.sk
myfit.sk1
Output
XploRe version 4.2a (Apr 28 1999)
Contents of _tmp
[1,] "Welcome to XploRe!"
Contents of dim
[1,] 2
Contents of b
[1,] 0.9967 0
[2,] 2.0131 0
[3,] 3.005 0
[4,] 4.0134 0
[5,] 0 5.0093
[6,] 0 5.9997
[7,] 0 6.9937
Contents of sk
[1,] 0 0
[2,] 0.423 0
Contents of sk1
[1,] 0 0
Leaving quantlet dpls

```

**Figure 2:** A `dpls`-session in XproRe

All matrices are structured in a comparable way. The number of rows are supposed to correspond with the number of variables and the number of columns should be identical with the corresponding number of observations. If the following model is taken as an example, which contains 41 manifest variables with 74 observations, then the matrix `y` has the shape 41x74. But the design matrix `dy` which contains the available connections between every manifest and every latent variable ("1" for a

connection and "0" for none) must have the shape 41x5 because this model contains 5 latent variables. One can observe the same structure in the matrix  $w$  with the difference that the matrix could contain at the spots of "1" any other value which should be used as starting weight by the algorithm.

The  $d$  and  $dl$  matrices are squared. The rows and the columns stand for latent variables. And since the following model is designed with connections all leading to the fifth variable, all rows are filled with noughts except of the last. This row describes which of the variables are connected to the fifth variable. With the same logic one have to decode the output variable  $sk$  which contains the unlagged path coefficients in the first part followed by the lagged ones.

The last tool for a dpls-session is the module *redun*. The following overview shows shortly the usage of that programme:

*Usage:*        {red, redm}=redun(b, sk, lk, skl, y)

*Input:*

b	a matrix of loadings
sk	a matrix of path coefficients
lk	a matrix of latent variables
skl	a matrix of lagged path coefficients
y	a matrix of manifest variables (indicators)

*Output:*

Red	a scalar of single redundancy value
Redm	a vector of redundancy values

A simple printout of an dpls-output can be found below.

Contents of wg	[6,] 0 5.9997
[1,] 0.032821 0	[7,] 0 6.9937
[2,] 0.066113 0	
[3,] 0.099959 0	Contents of sk
[4,] 0.13301 0	[1,] 0 0
[5,] 0 0.045413	[2,] 0.423 0
[6,] 0 0.054268	
[7,] 0 0.063904	Contents of skl
	[1,] 0 0
Contents of b	[2,] 0.856 0
[1,] 0.9967 0	
[2,] 2.0131 0	
[3,] 3.005 0	Contents of iter
[4,] 4.0134 0	[1,] 2
[5,] 0 5.0093	

## **3 EXAMPLE: A DYNAMIC LATENT VARIABLE MODEL FOR GERMAN SHARE PRICES**

### **3.1 The General Path Model**

The main purpose of DPLS in XploRe is the construction of general variables which optimally represents the dynamics of corresponding set of numerous indicators with their individual but similar dynamics. Indicators or manifest variables of this sort are share prices. Share price indexes are weighted sums of individual share prices. Weights are returns or quantities of stocks purchased, e.g. But by usage of dynamic path models the weights will be estimated as coefficients within the latent variable model. Such a coefficient represents a sort of importance of the individual share price within the construct of the latent variable in the context of the whole model.

The question to be answered by this specific model is, whether a certain construct of share prices can show the dynamic dependence of share prices on economic indicators. This construct could then be used as a kind of new share price index.

The challenge is to specify a dynamic model that represents the most important economic relationship of the index as a stable representant of the set of share prices under consideration.

Besides the latent share price variable (SP), the model under consideration contains the LV "Labour Market" (LM), "Money Market" (MM), "Domestic Economic Performance" (DP) and "Foreign Market" (FM).

The table next page shows the manifest variable belonging to these LVs. The figure in section 3.2 shows the relationships selected: The latent share price variable SP is assumed to statistically depend on LM, MM, DP, and FM. Furthermore a first-order auto-regression of SP and a first-order lagged dependency on LM is supposed. This selection is the result of some previous empirical pilot studies eliminating further lagged relationships.

## Manifest Variables and Sources of Data

	<b>Domestic Performance:</b>	<b>Origin of Data:</b>	<b>Unit:</b>
1	Incoming Orders (Processing Business)	Monthly Reports DBBK without Table 4	1991=100
2	Incoming Orders (Construction Industry)	Monthly Reports DBBK without Table 4	1991=100
3	Production (Manufacturing Business)	Monthly Reports DBBK without Table 4	1991=100
4	Production (Processing Business)	Monthly Reports DBBK without Table 4	1991=100
5	Commodity Trade (Exportation)	Zahlungsbilanzstatistik DBBK	Billion DM

	<b>Foreign Market</b>	<b>Origin of Data:</b>	<b>Unit:</b>
6	Dow Jones Industrial Average	Monthly Reports DBBK without Table 4	Index
7	Commodity Trade (Importation)	Zahlungsbilanzstatistik DBBK	1991=100
8	US\$ against 18 Industrial Countries	Monthly Reports DBBK without Table 4	1972=100
9	Incoming Orders from Foreign Countries (Proces. Bus.)	Monthly Reports DBBK without Table 4	1991=100
10	Discount Rate USA	Monthly Reports DBBK without Table 4	% p.A.

	<b>Money Market</b>	<b>Origin of Data:</b>	<b>Unit:</b>
11	Money Supply (M3)	Monthly Reports DBBK without Table 4	Billion DM
12	Discount Rate DBBK	Monthly Reports DBBK without Table 4	% p.A.
13	DM against US\$	Monthly Reports DBBK without Table 4	1972=100

	<b>Labour Market</b>	<b>Origin of Data:</b>	<b>Unit:</b>
14	Gross Earnings	Zahlungsbilanzstatistik DBBK	Quantity
15	Number of Unemployed	Zahlungsbilanzstatistik DBBK	Quantity
16	Vacancies	Zahlungsbilanzstatistik DBBK	Quantity
17	Short-Time Workers	Zahlungsbilanzstatistik DBBK	Quantity

	<b>German Stock Prices</b>	<b>Origin of Data:</b>	<b>Unit:</b>
18	ALV	Deutsche Börse	DM
19	BAS	Deutsche Börse	DM
20	BAY	Deutsche Börse	DM
21	BMW	Deutsche Börse	DM
22	CBK	Deutsche Börse	DM
23	DAI	Deutsche Börse	DM
24	DBK	Deutsche Börse	DM
25	DGS	Deutsche Börse	DM
26	DRB	Deutsche Börse	DM
27	HEN3	Deutsche Börse	DM
28	HOE	Deutsche Börse	DM
29	KAR	Deutsche Börse	DM
30	LHA	Deutsche Börse	DM
31	LIN	Deutsche Börse	DM
32	MAN	Deutsche Börse	DM
33	MMW	Deutsche Börse	DM
34	PRS	Deutsche Börse	DM
35	RWE	Deutsche Börse	DM
36	SCH	Deutsche Börse	DM
37	SIE	Deutsche Börse	DM
38	THY	Deutsche Börse	DM
39	VEB	Deutsche Börse	DM
40	VIA	Deutsche Börse	DM
41	VOW	Deutsche Börse	DM

### 3.2 Empirical Results

Both models with levels and with differences had been estimated. Transformations such as differences or logarithms can easily be added within the XploRe environment. Despite the models on level base have produced some significant path coefficients and high redundancy we have finally decided for a model on first difference base.

The figure next page shows the path coefficients of the latent variables and the weights of the manifest variables. Dotted arrows indicate lagged dependencies. The differentiated latent share-price variable has only a very low autoregressive component (0.08). This is only 1/10 of the amount found for levels. The strongest dependency is that on the latent foreign-market variable (0.33). The weakest one is that on the domestic economic performance (-0.06). We have found a medium degree dependency on the latent money-market variable (0.26). Furthermore there is a significant first order lagged relationship with the latent labour market (0.18).

The weights can be interpreted as the partial contribution of the individual manifest variable within the block of indicators belonging to a latent variable. In this meaning, it is easy to see that some of the MVs can be dropped because of their negligible contribution, e.g. Importation, Gross Earnings and each of the share prices taken alone.

## 4. CONCLUSIONS

DPLS in XploRe is a powerful tool for modelling dynamic interrelations between blocks of variables represented by latent variables suitable for these blocks. XploRe supplies the computing environment for transforming and ordering the input data as well as for the further analysis and application of the DPLS output.

It is a certain limitation of the method that the user has to decide whether to use the levels of a variable or its differences if it is non-stationary. Therefore, it is our goal to find an approach to include into the modelling process tests for the co-integration of the latent variables. The result would be a sort of latent variable error correction model.

Furthermore, there will be the possibility of different lag orders of the dynamic terms in the next version of the program.

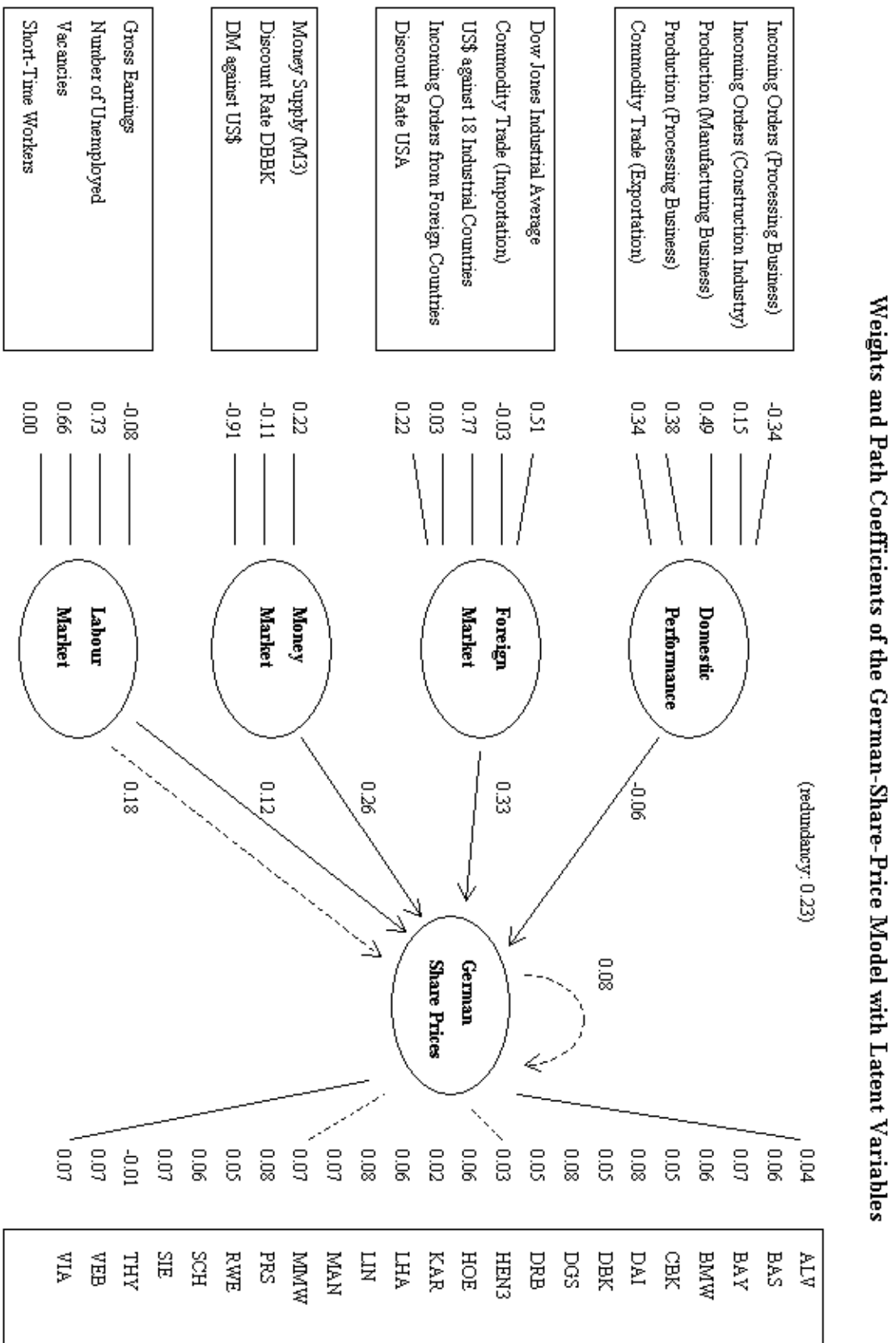


Figure 3: A model with German Share Prices

## References

- Geppert, F. (1996): Bearbeitung, Programmierung, Simulation und Anwendung eines PLS-Algorithmus für einfache dynamische Modelle mit latenten Variablen. Diploma thesis supervised by H. G. Strohe, Universität Potsdam.
- Haerdle, W; Klinke, S; Turlach B.A. (1995): XploRe: An Interactive Statistical Computing environment. Springer-Verlag, Berlin.
- Lohmoeller (1984): LVPLS 1.6 - Program Manual (Latent Variables Path Analysis with Partial Least Squares Estimation). Zentralarchiv für empirische Sozialforschung, Universität Köln.
- Lohmoeller, J.-B. (1989): Latent Variable Path Modelling with Partial Least Squares. Heidelberg.
- PC-ISP (1992): Users Guide and Command Descriptions, Datavision AG, Schweiz.
- Strohe, H.G. (1998): *A Heuristic Partial-Least-Squares Approach to Estimating Dynamic Path Models*, in: I. Balderjahn, R. Mathar, M. Schader (Eds.): Classification, Data Analysis, and Data Highways, Springer-Verlag, p. 192-204
- Strohe, H.G., Geppert, F. (1997): *DPLS - Algorithmus und Computerprogramm für dynamische Partial-Least-Squares-Modelle*. Statistische Diskussionsbeiträge Nr. 7, Universität Potsdam 1997, ISSN 0949-068X.
- Wold, H.: (1973): Nonlinear Iterative Partial Least Squares (NIPALS) Modelling - Some Current Development; in P.R. Krishnajah (Ed.), Multivariate Analysis (Vol. 3, p. 383-407), New York; Academic Press.
- Tenenhaus, M. (1998): La regression PLS. Éditions Technip, Paris.

## Appendix

### Output File (additional documented)

German Stock Prices, first differences, standardised, Model 6  
Dynamic PLS-Analysis, Output of the results:

Calculating Precision:

3.0000000

Number of Measurements:

74.000000

Weights W:

-0.33696932	0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00
0.14967273	0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00
0.48896715	0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00
0.38196337	0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00
0.34327808	0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00
0.00000000E+00	0.51273394	0.00000000E+00	0.00000000E+00	0.00000000E+00
0.00000000E+00	-0.28064229E-01	0.00000000E+00	0.00000000E+00	0.00000000E+00
0.00000000E+00	0.77328545	0.00000000E+00	0.00000000E+00	0.00000000E+00
0.00000000E+00	0.29578337E-01	0.00000000E+00	0.00000000E+00	0.00000000E+00
0.00000000E+00	0.00000000E+00	-0.22402388	0.00000000E+00	0.00000000E+00
0.00000000E+00	0.00000000E+00	0.11463123	0.00000000E+00	0.00000000E+00
0.00000000E+00	0.22258858	0.00000000E+00	0.00000000E+00	0.00000000E+00
0.00000000E+00	0.00000000E+00	0.90973961	0.00000000E+00	0.00000000E+00
0.00000000E+00	0.00000000E+00	0.00000000E+00	-0.81360690E-01	0.00000000E+00
0.00000000E+00	0.00000000E+00	0.00000000E+00	0.73182130	0.00000000E+00
0.00000000E+00	0.00000000E+00	0.00000000E+00	0.65653437	0.00000000E+00
0.00000000E+00	0.00000000E+00	0.00000000E+00	0.68048402E-02	0.00000000E+00
0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00	0.38965002E-01
...	...	...	...	...
0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00	0.71524501E-01

Loadings:

0.58465701	0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00
0.73239899	0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00
0.94316989	0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00
0.94336832	0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00
0.77453196	0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00
0.00000000E+00	0.54549420	0.00000000E+00	0.00000000E+00	0.00000000E+00
0.00000000E+00	0.12358556E-01	0.00000000E+00	0.00000000E+00	0.00000000E+00
0.00000000E+00	0.82903075	0.00000000E+00	0.00000000E+00	0.00000000E+00
0.00000000E+00	0.16803336	0.00000000E+00	0.00000000E+00	0.00000000E+00



0.00000000E+00	0.00000000E+00	-0.36637503	0.00000000E+00	0.00000000E+00
0.00000000E+00	0.00000000E+00	0.24562944	0.00000000E+00	0.00000000E+00
0.00000000E+00	0.33517450	0.00000000E+00	0.00000000E+00	0.00000000E+00
0.00000000E+00	0.00000000E+00	0.97804493	0.00000000E+00	0.00000000E+00
0.00000000E+00	0.00000000E+00	0.00000000E+00	0.14574070E-01	0.00000000E+00
0.00000000E+00	0.00000000E+00	0.00000000E+00	0.74758679	0.00000000E+00
0.00000000E+00	0.00000000E+00	0.00000000E+00	0.69109386	0.00000000E+00
0.00000000E+00	0.00000000E+00	0.00000000E+00	0.52735191E-01	0.00000000E+00
0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00	0.57629341
...	...	...	...	...
0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00	0.76450640

Path Coefficients (5\*5-Matrix (without Lags):

0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00
0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00
0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00
0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00
-0.56000002E-01	0.32600001	-0.25900000	0.12300000	0.00000000E+00

Path Coefficients. (5\*5-Matrix (with Lags):

0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00
0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00
0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00
0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00
0.00000000E+00	0.00000000E+00	0.00000000E+00	0.18400000	0.82999997E-01

Redundancy:

0.23219760

Redundancy Vector:

0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00
0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00
0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00
0.00000000E+00	0.00000000E+00	0.00000000E+00	0.00000000E+00
0.00000000E+00	0.16661556	0.25480044	0.27338761
0.24408513	0.15421984	0.36628929	0.19400176
0.31508613	0.15940319	0.59551671E-01	0.22974756
0.75901456E-01	0.27145961	0.33461976	0.33721462
0.33503371	0.29845610	0.21447843	0.14206910
0.27982733	0.33657333	0.20146195E-02	0.23468895
0.29321784			

Number of Iterations:

7.0000000