Panel Data estimation using DPD for Ox

BY JURGEN A. DOORNIK[1], MANUEL ARELLANO[2], AND STEPHEN BOND[3]

[1]*Nuffield College, Oxford*, http://www.nuff.ox.ac.uk/users/doornik/

[2]*CEMFI, Madrid*, http://www.cemfi.es/∼arellano/

[3]*Nuffield College, Oxford* and *IFS, London*

June 20, 2001

Contents

1 Introduction

DPD (Dynamic Panel Data) is a package for estimating dynamic panel data models. It also implements some of the static panel data estimators. DPD is a class written in Ox (see Doornik, 2001), and is used by writing small Ox programs which create and use an object of the DPD class. Some knowledge of Ox will be required when using DPD this way.

The DPD package can also be used interactively, similar to PcGive (Hendry and Doornik, 2001). This is the easiest way to use DPD. Required are Ox Professional, together with GiveWin 2.00 or newer; OxPack can be downloaded from the same location as DPD. Some examples are given below in §5.

The `DPD` class derives from the `Modelbase` class, which in turn derives from the the `Database` class to give easy loading of data sets and sample selection. An additional simulation class allows Monte Carlo experimentation with the facilities in the estimation class.

DPD is designed to handle unbalanced panels.

1.1 Disclaimer

This package is functional, but no warranty is given whatsoever. The most appropriate forum to discuss problems and issues related to the DPD package is the ox-users discussion group (subscription info and archiving is at `www.mailbase.ac.uk/lists/ox-users`). Please report suggestions for improvement to Jurgen Doornik (email for Manuel: Arellano@cemfi.es; for Steve: Steve.Bond@nuffield.ox.ac.uk email for Jurgen: Jurgen.Doornik@nuffield.ox.ac.uk).

There is also a Gauss version of DPD. The Ox version is a newly written implementation, which differs in the way it is used and the estimators which are implemented. See §6.1 for a list of numerical differences.

1.2 Availability and Citation

The DPD package is available for downloading through `http://www.nuff.ox.ac.uk/Users/ Doornik/`. DPD is written as 100% pure Ox code, and will also work on Unix platforms.

To facilitate replication and validation of empirical findings, please cite this documentation in all reports and publications involving the application of the `DPD` package.

This package must be cited whenever it is used.

1.3 Installation

(1) Make sure you have properly installed Ox version 2.10 or later. The `DPD` package does not work fully with earlier versions of Ox. Type `oxl` at the command prompt to check.
(2) Create a `dpd` subdirectory in the `ox\packages` folder and put `dpdox100.zip` in that subdirectory, then unzip `dpdox100.zip`.
(3) Read the `readme.txt` file for information on last minute changes.
(4) If Ox has been installed properly, this will allow using the `DPD` package from any directory. To use the package in your code, add the command

```
#import <packages/dpd/dpd>
```

at the top of all files which require it.

1.4 Running DPD code

To run the program in §3.2 under Windows 95/98/NT:

```
oxl abest1
```

You can also use OxRun to run the program in §3.2 under Windows 3.1/95/98/NT. In that case the output will appear in GiveWin, instead of on the MS-DOS console. DPD is written as 100% pure Ox code, and will also work on Unix platforms.

1.5 Main files

- `dpd.h` – the header file for the `DPD` class;
- `dpd.ox` – the source code with the `DPD` class;
- `dpd.oxo` – the compiled source code;
- `dpd.ps` – this document.
- `dpdsim.h` – the header file for the `DPDSim` class;
- `dpdsim.ox` – the source code with the `DPDSim` class;

Included sample data sets:

- `grunfeld.xls` – Grunfeld data in Excel spreadsheet format;
- `abdata.in7/bn7` – Arellano–Bond data in GiveWin format.

The remaining files are sample programs.

2 Data organization and model formulation

2.1 Data organization

The following data files can be read directly into a DPD object: GiveWin (.in7/.bn7), spreadsheet (Excel, Lotus), Stata, ASCII and Gauss (.dht/.dat). This is explained in the Ox manual.

In the remainder we shall use the word *individual* to denote the cross-sectional unit (which could be individual, firm, country, etc.).

Because DPD needs to recognize the structure of the data, and to allow for unbalanced panels, the data must be organized prior to analysis:

- Each variable is in a *column*.
- the columns are *ordered by individual*, and within individual by time.
- Each *row* refers to the same time period.
- a column with individual *index* is recommended (when omitted, the year variable is used to create such an index).
- There must be one column which denotes the *year*. This does not have to be the first column.
- An optional *period* column may be present when the data is not annual (i.e. the frequency is not one).
- An optional *group* column may be present to facilitate creating group dummies.
- *Missing values* are also called NaN (Not a Number, specified in Ox by a dot, or the constant `M_NAN`).

For example:

year	*index*	employment	wages	
1971	1	110	9	individual 1
1972	1	130	11	,,
1973	1	140	12	,,
1974	1	130	14	,,
1970	2	540	12	individual 2
1971	2	520	13	,,
1972	2	510	14	,,
1973	2	510	14	,,

If the data are not sorted, they can easily be sorted using code like:

```
#include <oxstd.h>
#import <database>

main()
{
    decl db = new Database();
    db.Load("data.in7");
    db.RenewBlock( sortbyc(db.GetAll(), <1,0>), 0);
    db.SaveIn7("newdata.in7");
}
```

This assumes that an individual index is in column 1, and a time index in column 0, so that the data are sorted by individual by time.

In general, once the data set is sorted, the DPD code works out where individuals start and finish by differencing the index variable. As shown here, when the index variable is differenced once, a non-zero value indicates the start of an individual, with the zeros indicating continuation:

year	*index*	empl	Δ*year*	Δ*index*	Δemp	
1971	1	110	.	-1	.	individual 1
1972	1	130	1	0	10	,,
1973	1	140	1	0	10	,,
1974	1	130	1	0	-10	,,
1970	4	540	-4	3	.	individual 4
1971	4	520	1	0	-20	,,
1972	4	510	1	0	-10	,,
1973	4	510	1	0	0	,,
1973	3	30	0	-1	.	individual 3
1974	3	20	1	0	-10	,,
1975	5	30	1	2	.	individual 5

When the index variable is missing, the DPD code will try to work out where an individual starts and ends using the year variable. This is done by differencing the year variable, and checking if the resulting value corresponds to the frequency of the data (for annual data: a one indicates the continuation of an individual). The drawback of this approach is that it excludes subsequent individuals where the year of the last observation on an individual is immediately after the year of the first observation on the next individual.

2.2 Model formulation

Model formulation is based on the names of variables, not the column index. The following steps are involved in model formulation:

- Create a DPD object.
- Load your data into the DPD database using the facilities of the Database class.
- Transform the data.
- Optionally use `SetIndex` to specify the data column with the index variable.
- Use `SetYear` to specify the data column with years.
- Optionally use `SetGroup` to specify the data column with groups.
- Use `Select` to formulate the model.
- A constant will be included by default, use `SetDummies` for a different set of dummies, for example:
  - none: `D_NONE`
  - constant term (the default): `D_CONSTANT`
  - time dummies: `D_TIME`
  - group dummies: `D_GROUP`
  - time dummies interacted with group dummies: `D_TIMEGROUP`
  - individual dummies: `D_INDIVIDUAL`

- Use the `Gmm` function to specify GMM instruments for the transformed equations.
- For combined levels and differences/deviations estimation use the `GmmLevel` function to specify GMM instruments for the level equations.
- Use `SetTransform` to select a transformation method for estimation:
  - first differences (the default): `T_DIFFERENCES`
  - orthogonal transformations: `T_DEVIATIONS`
  - levels (no transformation): `T_NONE`
  - within group (deviations from individual means): `T_WITHIN`
  - between groups: `T_BETWEEN`
  - GLS: `T_GLS`
- Use `SetMethod` to change estimation method, for example:
  - 1-step estimation (the default): `M_1STEP`
  - 2-step estimation: `M_2STEP`
  - Feasible GLS (static panels): `M_DPD_GLS`
  - Maximum likelihood (static panels): `M_DPD_ML`

  Robust 1-step estimates are automatically reported when doing 2-step estimation.
- By default, Wald tests on the regressors (except dummies) and the dummies are reported, as well as Sargan's test for overidentifying restrictions (the last not for robust 1-step estimation). Use `SetTest` to change this, or to add AR tests.
- Finally, use `Estimate` for estimation.

3 Dynamic panel data estimation

As the name suggests, DPD can be used to compute a variety of dynamic panel estimators, particularly the GMM-type estimators of Arellano and Bond (1991), Arellano and Bover (1995), and Blundell and Bond (1998). In addition, some of the Anderson and Hsiao (1982) methods can also be estimated.

The available estimators are:

	`SetMethod`	Example
OLS in levels (no instruments specified)	`M_1STEP`	`abest2.ox`
one-step IV estimation (Andersion–Hsiao IV)	`M_1STEP`	`abest2.ox`
one-step GMM estimation	`M_1STEP`	
one-step estimation with robust std. errors	`M_2STEP`	`abest1.ox`
two-step estimation	`M_2STEP`	`abest1.ox`

3.1 Econometric methods

The general model that can be estimated with DPD is a single equation with individual effects of the form:

$$y_{it} = \sum_{k=1}^{p} \alpha_k y_{i(t-k)} + \beta'(L) x_{it} + \lambda_t + \eta_i + v_{it}, \quad t = q+1, ..., T_i;\ i = 1, ..., N,$$

where η_i and λ_t are respectively individual and time specific effects, x_{it} is a vector of explanatory variables, $\beta(L)$ is a vector of associated polynomials in the lag operator and q is the maximum lag length in the model. The number of time periods available on the ith individual, T_i, is small and the number of individuals, N, is large. Identification of the model requires restrictions on the serial correlation properties of the error term v_{it} and/or on the properties of the explanatory variables x_{it}. It is assumed that if the error term was originally autoregressive, the model has been transformed so that the coefficients α's and β's satisfy some set of common factor restrictions. Thus only serially uncorrelated or moving average errors are explicitly allowed. The v_{it} are assumed to be independently distributed across individuals with zero mean, but arbitrary forms of heteroskedasticity across units and time are possible. The x_{it} may or may not be correlated with the individual effects η_i, and for each of theses cases they

may be strictly exogenous, predetermined or endogenous variables with respect to v_{it}. A case of particular interest is where the levels x_{it} are correlated with η_i but where Δx_{it} (and possibly Δy_{it}) are uncorrelated with η_i; this allows the use of (suitably lagged) Δx_{is} (and possibly Δy_{is}) as instruments for equations in levels.

The $(T_i - q)$ equations for individual i can be written conveniently in the form:

$$\boldsymbol{y}_i = \boldsymbol{W}_i\boldsymbol{\delta} + \boldsymbol{\iota}_i\eta_i + \boldsymbol{v}_i,$$

where $\boldsymbol{\delta}$ is a parameter vector including the α_k's, the β's and the λ's, and $\boldsymbol{W}_i$ is a data matrix containing the time series of the lagged dependent variables, the x's and the time dummies. Lastly, $\boldsymbol{\iota}_i$ is a $(T_i - q) \times 1$ vector of ones. DPD can be used to compute various linear GMM estimators of $\boldsymbol{\delta}$ with the general form:

$$\widehat{\boldsymbol{\delta}} = \left[\left(\sum_i \boldsymbol{W}_i^{*\prime}\boldsymbol{Z}_i\right)\boldsymbol{A}_N\left(\sum_i \boldsymbol{Z}_i^{\prime}\boldsymbol{W}_i^{*}\right)\right]^{-1}\left(\sum_i \boldsymbol{W}_i^{*\prime}\boldsymbol{Z}_i\right)\boldsymbol{A}_N\left(\sum_i \boldsymbol{Z}_i^{\prime}\boldsymbol{y}_i^{*}\right),$$

where

$$\boldsymbol{A}_N = \left(\frac{1}{N}\sum_i \boldsymbol{Z}_i^{\prime}\boldsymbol{H}_i\boldsymbol{Z}_i\right)^{-1},$$

and $\boldsymbol{W}_i^*$ and $\boldsymbol{y}_i^*$ denote some transformation of $\boldsymbol{W}_i$ and $\boldsymbol{y}_i$ (e.g. levels, first differences, orthogonal deviations, combinations of first differences (or orthogonal deviations) and levels, deviations from individual means). $\boldsymbol{Z}_i$ is a matrix of instrumental variables which may or may not be entirely internal, and $\boldsymbol{H}_i$ is a possibly individual specific weighting matrix.

If the number of columns of $\boldsymbol{Z}_i$ equals that of $\boldsymbol{W}_i^*$, $\boldsymbol{A}_N$ becomes irrelevant and $\widehat{\boldsymbol{\delta}}$ reduces to

$$\boldsymbol{\delta} = \left(\sum_i \boldsymbol{Z}_i^{\prime}\boldsymbol{W}_i^{*}\right)^{-1}\left(\sum_i \boldsymbol{Z}_i^{\prime}\boldsymbol{y}_i^{*}\right).$$

In particular, if $\boldsymbol{Z}_i = \boldsymbol{W}_i^*$ and the transformed $\boldsymbol{W}_i$ and $\boldsymbol{y}_i$ are deviations from individual means or orthogonal deviations[1], then $\widehat{\boldsymbol{\delta}}$ is the within groups estimator. As another example, if the transformation denotes first differences, $\boldsymbol{Z}_i = \boldsymbol{I}_{T_i} \otimes \boldsymbol{x}_i^{\prime}$ and $\boldsymbol{H}_i = \widehat{\boldsymbol{v}}_i^*\widehat{\boldsymbol{v}}_i^{*\prime}$, where the $\widehat{\boldsymbol{v}}_i^*$ are some consistent estimates of the first differenced residuals, then $\widehat{\boldsymbol{\delta}}$ is the generalised three stage least squares estimator of Chamberlain (1984) (in Griliches and Intriligator, 1984). These two estimators require the x_{it} to be strictly exogenous with respect to v_{it} for consistency. In addition, the within groups estimator can only be consistent as $N \to \infty$ for fixed T if $\boldsymbol{W}_i^*$ does not contain lagged dependent variables and all the explanatory variables are strictly exogenous.

When estimating dynamic models, we shall therefore typically be concerned with transformations that allow the use of lagged endogenous (and predetermined) variables as instruments in the transformed equations. Efficient GMM estimators will typically exploit a different number of instruments in each time period. Estimators of this type are discussed in Arellano (1988), Arellano and Bond (1991), Arellano and Bover (1995) and Blundell and Bond (1998). DPD can be used to compute a range of linear GMM estimators of this type.

Where there are no instruments available that are uncorrelated with the individual effects η_i, the transformation must eliminate this component of the error term. The first difference and orthogonal deviations transformations are two examples of transformations that eliminate η_i from the transformed error term, without at the same time introducing all lagged values of the distrubances v_{it} into the transformed error term.[2] Hence these transformations allow the use of suitably lagged endogenous (and predetermined) variables as instruments. For example, if the panel is balanced, $p = 1$, there are no explanatory variables nor time effects, the v_{it} are serially uncorrelated, and the initial conditions y_{i1} are uncorrelated with v_{it} for $t = 2, ..., T$, then using first differences we have:

[1] Orthogonal deviations, as proposed by Arellano (1988) and Arellano and Bover (1995), express each observation as the deviation from the average of *future* observations in the sample for the same individual, and weight each deviation to standardise the variance, i.e.

$$x_{it}^* = \left(x_{it} - \frac{x_{i(t+1)} + ... + x_{iT}}{T-t}\right)\left(\frac{T-t}{T-t+1}\right)^{1/2} \text{ for } t = 1, ..., T-1.$$

If the original errors are serially uncorrelated and homoskedastic, the transformed errors will also be serially uncorrelated and homoskedastic.

[2] There are many other transformations which share these properties. See Arellano and Bover (1995) for further discussion.

Equations	Instruments available
$\Delta y_{i3} = \alpha \Delta y_{i2} + \Delta v_{i3}$	y_{i1}
$\Delta y_{i4} = \alpha \Delta y_{i3} + \Delta v_{i4}$	y_{i1}, y_{i2}
$\vdots$	$\vdots$
$\Delta y_{iT} = \alpha \Delta y_{i(T-1)} + \Delta v_{iT}$	$y_{i1}, y_{i2}, ..., y_{i(T-2)}$

In this case $\boldsymbol{y}_i^* = (\Delta y_{i3}, ..., \Delta y_{iT})'$, $\boldsymbol{W}_i^* = (\Delta y_{i2}, ..., \Delta y_{i(T-1)})'$ and

$$\boldsymbol{Z}_i = \boldsymbol{Z}_i^D = \begin{pmatrix} y_{i1} & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & y_{i1} & y_{i2} & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & y_{i1} & y_{i2} & \cdots & y_{i(T-2)} \end{pmatrix}$$

Notice that precisely the same instrument set would be used to estimate the model in orthogonal deviations. Where the panel is unbalanced, for individuals with incomplete data the rows of Z_i corresponding to the missing equations are deleted, and missing values in the remaining rows are replaced by zeros.

In DPD we call one-step estimates those which use some known matrix as the choice for H_i. For a first-difference procedure, the one-step estimator uses

$$\boldsymbol{H}_i = \boldsymbol{H}_i^D = \frac{1}{2} \begin{pmatrix} 2 & -1 & \cdots & & 0 \\ -1 & 2 & \cdots & & 0 \\ \vdots & \vdots & & & \vdots \\ \cdot & \cdot & & & -1 \\ 0 & 0 & \cdots & -1 & 2 \end{pmatrix},$$

while for a levels or orthogonal deviations procedure the one-step estimator sets H_i to an identity matrix. If the v_{it} are heteroskedastic, a two-step estimator which uses

$$\boldsymbol{H}_i = \widehat{\boldsymbol{v}}_i^* \widehat{\boldsymbol{v}}_i^{*\prime},$$

where $\widehat{\boldsymbol{v}}_i^*$ are one-step residuals, is more efficient (cf. White, 1982). DPD produces both one-step and two-step GMM estimators, with asymptotic variance matrices that are heteroskedasticity-consistent in both cases. Users should note that, particularly when the v_{it} are heteroskedastic, simulations suggest that the asymptotic standard errors for the two-step estimators can be a poor guide for hypothesis testing in typical sample sizes. In these cases, inference based on asymptotic standard errors for the one-step estimators seems to be more reliable (see §3.4, Arellano and Bond, 1991, and Blundell and Bond, 1998 for further discussion.

In models with explanatory variables, $\boldsymbol{Z}_i$ may consist of sub-matrices with the block diagonal form illustrated above (exploiting all or part of the moment restrictions available), concatenated to straightforward one-column instruments. A judicious choice of the $\boldsymbol{Z}_i$ matrix should strike a compromise between prior knowledge (from economic theory and previous empirical work), the characteristics of the sample and computer limitations (see Arellano and Bond (1991) for an extended discussion and illustration). For example, if a predetermined regressor x_{it} correlated with the individual effect, is added to the model discussed above, i.e.

$$\begin{aligned} E(x_{it} v_{is}) &= 0 \text{ for } s \geq t \\ &\neq 0 \text{ otherwise} \\ E(x_{it} \eta_i) &\neq 0 \end{aligned}$$

then the corresponding optimal Z_i matrix is given by

$$\boldsymbol{Z}_i = \begin{pmatrix} y_{i1} & x_{i1} & x_{i2} & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & y_{i1} & y_{i2} & x_{i1} & x_{i2} & x_{i3} & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & y_{i1} & \cdots & y_{i(T-2)} & x_{i1} & \cdots & x_{i(T-1)} \end{pmatrix}$$

Where the number of columns in $\boldsymbol{Z}_i$ is very large, computational considerations may require those columns containing the least informative instruments to be deleted. Even when computing speed is not an issue, it may be advisable not to use the whole history of the series as instruments in the later cross-sections. For a given cross-sectional sample size (N), the use of too many instruments may result in (small sample) overfitting biases. When overfitting results from the number of time periods (T) becoming large relative to the number of individuals (N), and there are no endogenous regressors present, these GMM estimators are biased towards within groups, which is not a serious concern since the within groups estimator is itself consistent for models with predetermined variables as T becomes large (see Alvarez and Arellano, 1998). However, in models with endogenous regressors, using too many instruments in the later cross-sections could result in seriously biased estimates. This possibility can be investigated in practice by comparing the GMM and within groups estimates.

The assumption of no serial correlation in the v_{it} is essential for the consistency of estimators such as those considered in the previous examples, which instrument the lagged dependent variable with further lags of the same variable. Thus DPD reports tests for the absence of first-order and second-order serial correlation in the first-differenced residuals. If the disturbances v_{it} are not serially correlated, there should be evidence of significant negative first order serial correlation in differenced residuals (i.e. $\widehat{v}_{it} - \widehat{v}_{i,t-1}$), and no evidence of second order serial correlation in the differenced residuals. These tests are based on the standardised average residual autocovariances which are asymptotically $N(0,1)$ variables under the null of no autocorrelation. The tests reported are based on estimates of the residuals in first differences, even when the estimator is obtained using orthogonal deviations.[3] More generally, Sargan tests of overidentifying restrictions are also reported. That is, if $\boldsymbol{A}_N$ has been chosen optimally for any given $\boldsymbol{Z}_i$, the statistic

$$S = \left(\sum_i \widehat{\boldsymbol{v}}_i^{*\prime} \boldsymbol{Z}_i \right) \boldsymbol{A}_N \left(\sum_i \boldsymbol{Z}_i^{\prime} \widehat{\boldsymbol{v}}_i^{*} \right) \tag{1}$$

is asymptotically distributed as a chi-square with as many degrees of freedom as overidentifying restrictions, under the null hypothesis of the validity of the instruments. Note that only the Sargan test based on the two-step GMM estimator is heteroskedasticity-consistent. Again, Arellano and Bond (1991) provide a complete discussion of these procedures.

Where there are instruments available that are uncorrelated with the individual effects η_i, these variables can be used as instruments for the equations in levels. Typically this will imply a set of moment conditions relating to the equations in first differences (or orthogonal deviations) and a set of moment conditions relating to the equations in levels, which need to be combined to obtain the efficient GMM estimator.[4] For example, if the simple AR(1) model considered earlier is mean-stationary, then the first differences Δy_{it} will be uncorrelated with η_i, and this implies that $\Delta y_{i(t-1)}$ can be used as instruments in the levels equations (see Arellano and Bover, 1995 and Blundell and Bond, 1998 for further discussion). In addition to the instruments available for the first-differenced equations that were described earlier, we then have:

Equations	Instruments available
$y_{i3} = \alpha y_{i2} + \eta_i + v_{i3}$	Δy_{i2}
$y_{i4} = \alpha y_{i3} + \eta_i + v_{i4}$	Δy_{i3}
$\vdots$	$\vdots$
$y_{iT} = \alpha y_{i(T-1)} + \eta_i + v_{iT}$	$\Delta y_{i(T-1)}$

Notice that no instruments are available in this case for the first levels equation (i.e. $y_{i2} = \alpha y_{i1} + \eta_i + v_{i2}$), and that using further lags of Δy_{is} as instruments here would be redundant, given the instruments that are being used for the equations in first differences. In a balanced panel, we could use only the last levels equation (i.e. $y_{iT} = \alpha y_{i(T-1)} + \eta_i + v_{iT}$), where $(\Delta y_{i2}, \Delta y_{i3}, ..., \Delta y_{i(T-1)})$ would all be valid instruments; however this approach does not extend conveniently to unbalanced panels.

[3] Although the validity of orthogonality conditions is not affected, the transformation to orthogonal deviations can induce serial correlation in the transformed error term if the v_{it} are serially uncorrelated but heteroskedastic.

[4] In special cases it may be efficient to use only the equations in levels; for example, in a model with no lagged dependent variables and all regressors strictly exogenous and uncorrelated with individual effects.

In this case, we use $y_i^* = (\Delta y_{i3}, ..., \Delta y_{iT}, y_{i3}, ..., y_{iT})'$, $W_i^* = (\Delta y_{i2}, ..., \Delta y_{i(T-1)}, y_{i2}, ..., y_{i(T-1)})'$ and

$$\boldsymbol{Z}_i = \begin{pmatrix} \boldsymbol{Z}_i^{\boldsymbol{D}} & 0 & \cdots & 0 \\ 0 & \Delta y_{i2} & \cdots & 0 \\ . & . & & . \\ 0 & 0 & \cdots & \Delta y_{i(T-1)} \end{pmatrix}$$

where $\boldsymbol{Z}_i^D$ is the matrix of instruments for the equations in first differences, as described above. Again $\boldsymbol{Z}_i$ would be precisely the same if the transformed equations in $\boldsymbol{y}_i^*$ and $\boldsymbol{W}_i^*$ were in orthogonal deviations rather than first differences. In models with explanatory variables, it may be that the levels of some variables are uncorrelated with η_i, in which case suitably lagged levels of these variables can be used as instruments in the levels equations, and in this case there may be instruments available for the first levels equation.

For the system of equations in first differences and levels, the one-step estimator computed in DPD uses the weighting matrix

$$\boldsymbol{H}_i = \begin{pmatrix} \boldsymbol{H}_i^D & 0 \\ 0 & \frac{1}{2} I_i \end{pmatrix}$$

where $\boldsymbol{H}_i^D$ is the weighting matrix described above for the first differenced estimator, and $\boldsymbol{I}_i$ is an identity matrix with dimension equal to the number of levels equations observed for individual i. For the system of equations in orthogonal deviations and levels, the one-step estimator computed in DPD sets $\boldsymbol{H}_i$ to an identity matrix with dimension equal to the total number of equations in the system for individual i. In both cases the corresponding two-step estimator uses $\boldsymbol{H}_i = \widehat{\boldsymbol{v}}_i^* \widehat{\boldsymbol{v}}_i^{*\prime}$. We adopt these particular one-step weighting matrices because they are equivalent in the following sense: for a balanced panel where all the available linear moment restrictions are exploited (i.e. no columns of $\boldsymbol{Z}_i$ are omitted for computational or small sample reasons), the associated one-step GMM estimators are numerically identical, regardless of whether the first difference or orthogonal deviations transformation is used to construct the system. Notice though that the one-step estimator is asymptotically inefficient relative to the two-step estimator for both of these systems, even if the v_{it} are homoskedastic.[5] Again simulations have suggested that asymptotic inference based on the one-step versions may be more reliable than asymptotic inference based on the two-step versions, even in moderately large samples (see §3.4 and Blundell and Bond, 1998).

The validity of these extra instruments in the levels equations can be tested using the Sargan statistic provided by DPD. Since the set of instruments used for the equations in first differences (or orthogonal deviations) is a strict subset of that used in the system of first-differenced (or orthogonal deviations) and levels equations, a more specific test of these additional instruments is a Difference Sargan test which compares the Sargan statistic for the system estimator and the Sargan statistic for the corresponding first-differenced (or orthogonal deviations) estimator. Another possibility is to compare these estimates using a Hausman specification test, which can be computed here by including another set of regressors that take the value zero in the equations in first differences (or orthogonal deviations), and reproduce the levels of the right hand side variables for the equations in levels.[6] The test statistic is then a Wald test of the hypothesis that the coefficients on these additional regressors are jointly zero. Full details of these test procedures can be found in Arellano and Bond (1991) and Arellano (1995).

[5] With levels equations included in the system, the optimal weight matrix depends on unknown parameters (for example, the ratio of var(η_i) to var(v_{it})) even in the homoskedastic case.

[6] Thus in the AR(1) case described above we would have

$$\boldsymbol{W}_i^* = \begin{pmatrix} 0 & ... & 0 & y_{i2} & ... & y_{i(T-1)} \\ \Delta y_{i2} & ... & \Delta y_{i(T-1)} & y_{i2} & ... & y_{i(T-1)} \end{pmatrix}'.$$

3.2 Examples on dynamic panel data estimation

The code below (provided as `abest1.ox`) estimates column (b) of table 4 in Arellano and Bond (1991), using the data set `abdata.in7`:

abest1.ox

```
#include <oxstd.h>
#import <packages/dpd/dpd>

main()
{
    decl dpd = new DPD(), time = timer(), x;

    dpd.Load("abdata.in7");                   // load data
    dpd.SetYear("YEAR");     // specify columns with years
    dpd.SetOptions(FALSE);         // no robust std.errors

    dpd.Select(Y_VAR, {"n", 0, 2});     // formulate model
    dpd.Select(X_VAR, {"w", 0, 1, "k", 0, 0, "ys", 0, 1});
    dpd.Select(I_VAR, {"w", 0, 1, "k", 0, 0, "ys", 0, 1});

    dpd.Gmm("n", 2, 99);            // GMM-type instrument
    dpd.SetDummies(D_CONSTANT + D_TIME);// specify dummies
    dpd.SetTest(1, 2);// specification,Sargan,AR 1-2 tests
    dpd.Estimate();                   // 1-step estimation

    print("\n\n***** Arellano & Bond (1991), Table 4 (b)");
    dpd.SetMethod(M_2STEP);
    dpd.Estimate();                   // 2-step estimation
                          // this gives table 4, column (b)
    print("\ntime: ", timespan(time), "\n");
    delete dpd;                      // finished with object
}
```

Which generates output (one-step results are not shown; the full output is in `abest1.out`):

part of abest1.out

```
***** Arellano & Bond (1991), Table 4 (b)
DPD( 2) Modelling n by 1 and 2 step

                  ----   2-step estimation using DPD ----
                  Coefficient  Std.Error  t-value  t-prob
Dn(-1)               0.474151    0.08530     5.56   0.000
Dn(-2)              -0.0529675   0.02728    -1.94   0.053
Dw                   -0.513205   0.04935    -10.4   0.000
Dw(-1)               0.224640    0.08006     2.81   0.005
Dk                   0.292723    0.03946     7.42   0.000
Dys                  0.609775     0.1085     5.62   0.000
Dys(-1)             -0.446373     0.1248    -3.58   0.000
Constant             0.0105090   0.007251     1.45   0.148
T1980               0.00363321   0.01273    0.285   0.775
T1981               -0.0509621   0.01371    -3.72   0.000
T1982               -0.0321490   0.01399    -2.30   0.022
T1983               -0.0123558   0.01284   -0.962   0.336
T1984               -0.0207295   0.01368    -1.52   0.130

sigma                0.116243  sigma^2           0.01351243
sigma levels        0.08219621
RSS               8.0804358435  TSS              12.599978399
no. of observations       611  no. of parameters          13
Warning: standard errors are unreliable!
```

(continued)

```
Transformation used:     first differences
Transformed instruments:  w  w(-1)  k  ys  ys(-1)
Level instruments:        Dummies  Gmm(n,2,99)

constant:                 yes  time dummies:              5
number of individuals     140 (derived from year)
longest time series         6 [1979 - 1984]
shortest time series        4 (unbalanced panel)

Wald (joint):    Chi^2(7) =     372.0 [0.000] **
Wald (dummy):    Chi^2(6) =     26.90 [0.000] **
Wald (time):     Chi^2(6) =     26.90 [0.000] **
Sargan test:    Chi^2(25) =     30.11 [0.220]
AR(1) test:        N(0,1) =    -2.428 [0.015] *
AR(2) test:        N(0,1) =   -0.3325 [0.739]
```

These results are identical to those in Arellano and Bond (1991), except for the AR(2) test (denoted m_2 in the paper), which is explained in the footnote on Page 28.

Note that, to replicate the published results, we switched off robust standard errors here. Starting with DPD version 1.2, robust standard errors for two-step GMM are available, based on a small-sample correction by Windmeijer (2000).

Further examples are:

- `abest2.ox`, which replicates Table 5 from Arellano and Bond (1991). The basic specification is:

```
dpd.Select(Y_VAR, {"n", 0, 0});      // formulate model
dpd.Select(X_VAR, {"n", 1, 2, "w", 0, 1, "k", 0, 2, "ys", 0, 2});
dpd.SetDummies(D_CONSTANT + D_TIME);  // time, constant
```

  – OLS estimation:

```
dpd.SetTransform(T_NONE);         // estimate in levels
dpd.Estimate();                    // 1-step estimation
```

  – within-group estimation (both direct and using orthogonal deviations);

```
dpd.SetTransform(T_WITHIN); // proper within estimation
dpd.Estimate();                    // 1-step estimation
// and:
dpd.SetTransform(T_DEVIATIONS);      // estimate within
dpd.Estimate();                   // 1-step estimation
```

    Arellano and Bond (1991) implemented the within-group estimator as OLS after applying the orthogonal deviations estimator. In unbalanced panels this is slightly different from estimation in deviation from the mean of each individual (but asymptotically identical). DPD for Ox also has the within-groups estimation as an option (`abest2.ox` estimates both forms). The orthogonal deviations method has the benefit that robust standard errors are available when using two-step estimation.

  – Anderson–Hsiao-type estimates, using $\Delta n_{i,t-3}$ in addition to the regressors as non-GMM type instrument (`I_VAR`):

```
dpd.Select(I_VAR, {"n", 2, 3, "w", 0, 1, "k", 0, 2, "ys", 0, 2});
```

    Or entering all regressors as non-GMM type instruments in differences (`I_VAR`) and $n_{i,t-3}$ in levels (i.e. untransformed; `IL_VAR`):

```
dpd.Select(I_VAR, {"n", 2, 2, "w", 0, 1, "k", 0, 2, "ys", 0, 2});
dpd.Select(IL_VAR, {"n", 3, 3});
```

- `abest3.ox`, which replicates Table 4 from Arellano and Bond (1991). This has an analogue to column (c) which uses several GMM-type instruments:

$$\text{diag}(n_{i,1} \cdots n_{i,t-2} w_{i,t-3} w_{i,t-2} k_{i,t-3} k_{i,t-2}), \quad t = 3, \ldots T,$$

which is formulated as

```
dpd.Gmm("n", 2, 99);
dpd.Gmm("w", 2, 3);
dpd.Gmm("k", 2, 3);
```

Column (c) cannot be replicated because sales and stocks are not in the data set.

- `abest4.ox` repeats `abest1.ox`, but now using orthogonal deviations estimation:

```
dpd.SetTransform(T_DEVIATIONS);   // orthogonal deviations
```

Some indications of the time it takes to run these programs: the timings for most of the `abest` programs is less than a second on a 500 Mhz Pentium III, while the program of the next section, `bbest1`, takes about 3 seconds (running Windows NT 4.0).

3.3 Examples on combined dynamic panel data estimation

The program `bbest1.ox` has examples using the combined GMM-SYS estimator proposed in Arellano and Bover (1995) and Blundell and Bond (1998). In this type of estimation (called system estimator by Blundell and Bond, 1998) the level equations are stacked on top of the transformed equations. The GMM-type instruments for the differenced equations

$$\text{diag}(n_{i,1} \cdots n_{i,t-2} w_{i,1} \cdots w_{i,t-2} k_{i,1} \cdots k_{i,t-2}),$$

are formulated as:

```
dpd.Gmm("n", 2, 99);
dpd.Gmm("w", 2, 99);
dpd.Gmm("k", 2, 99);
```

The GMM-style instruments in the levels equation are the lagged differences:

$$\text{diag}(\Delta n_{i,t-1} \Delta w_{i,t-1} \Delta k_{i,t-1}),$$

formulated in `bbest1.ox` as:

```
dpd.GmmLevel("n", 1, 1);  // GMM instruments for levels
dpd.GmmLevel("w", 1, 1);
dpd.GmmLevel("k", 1, 1);
```

Table 1 Blundell & Bond Table 4: employment equations.

	1979-84 GMM-DIF		1979-84 GMM-SYS		1976-84 GMM-DIF		1976-84 GMM-SYS	
$n_{i,t-1}$	0.5393	(0.151)	0.9360	(0.062)	0.7075	(0.084)	0.8714	(0.044)
$w_{i,t}$	−1.2356	(0.373)	−1.0582	(0.220)	−0.7088	(0.117)	−0.7811	(0.116)
$w_{i,t-1}$	0.5405	(0.178)	0.8326	(0.158)	0.5000	(0.111)	0.5121	(0.167)
$k_{i,t}$	0.8291	(0.346)	0.9204	(0.217)	0.4660	(0.101)	0.4688	(0.071)
$k_{i,t-1}$	−0.7355	(0.294)	−0.8945	(0.236)	−0.2151	(0.086)	−0.3560	(0.072)
m_1	−3.15	[0.00]**	−3.72	[0.00]**	−5.60	[0.00]**	−5.98	[0.00]**
m_2	0.78	[0.44]	0.85	[0.40]	−0.14	[0.89]	−0.17	[0.87]
Sargan	13.28	[0.97]	38.04	[0.42]	88.80	[0.21]	111.6	[0.20]

Except for column 3, Table 4 of Blundell and Bond (1998) cannot be exactly replicated for the following reasons:

(1) sub-sample estimation was not implemented correctly, using more lagged information than appropriate. This affects the first two columns.
(2) The $\boldsymbol{H}_i$ matrix was set to the identity matrix. Subsequently, it has been decided to use the same matrix for differenced estimation whether not combined or combined (1 on the diagonal, -1/2 along the diagonal). This affects column 2 and 4.

(3) In Blundell and Bond (1998), the differenced dummies were used as instruments in the transformed equations. This affects column 2 and 4.

Using the program `bbest1.ox` we find the full-sample results listed in the last two columns of Table 1. The coefficients are for the one-step estimates, with in parentheses the robust standard errors. The Sargan test is the two-step version.

3.4 A simulation example

A Monte Carlo experiment generally consists of three components:

(1) data generation process (DGP),
(2) model (and test statistics) to simulate,
(3) Monte Carlo replications with accumulation of results.

All those ingredients are readily available: for many dynamic panels data DGPs, we can use the `PcNaiveDgp` class; the third step (the actual experiment) can be done by deriving from the `Simulation` class. The middle step is provided by the `DPD` class.

The homoscedastic DGP in Arellano and Bond (1991) is:

$$\begin{aligned} y_{it} &= \alpha y_{i,t-1} + \beta x_{i1} + \eta_i + u_{it}, \quad \eta_i \sim \mathsf{N}[0,1], \quad i = 1,\ldots,N,\ t = 1,\ldots,T \\ u_{it} &= \phi u_{i,t-1} + v_{it}, \quad v_{it} \sim \mathsf{N}[0,1], \\ x_{it} &= \rho x_{i,t-1} + e_{it}, \quad e_{it} \sim \mathsf{N}[0,\sigma_e^2]. \end{aligned}$$

This can be seen as a simultaneous equations system with N equations, which allows us to use the `PcNaiveDgp` class to generate data.

The Monte Carlo experiment is implemented in the `DPDSim` class. The estimated model includes a constant term. The estimators can be summarized as:

	transformation	regressors	instruments	estimation
OLS	–	$\boldsymbol{y}_{i,-1}, \boldsymbol{x}_i, \boldsymbol{1}$		1-step
Within	within	$\boldsymbol{y}_{i,-1}, \boldsymbol{x}_i, \boldsymbol{1}$		1-step
GMM1	Δ	$\Delta\boldsymbol{y}_{i,-1}, \Delta\boldsymbol{x}_i, \boldsymbol{1}$	$\mathrm{diag}(y_{i,t-3}y_{i,t-2}), \Delta\boldsymbol{x}_i, \boldsymbol{1}$	1-step
GMM2	Δ	as GMM1	as GMM1	2-step
AHd	Δ	as GMM1	$\Delta\boldsymbol{y}_{i,t-2}, \Delta\boldsymbol{x}_i, \boldsymbol{1}$	1-step
AHl	Δ	as GMM1	$\boldsymbol{y}_{i,t-2}, \Delta\boldsymbol{x}_i, \boldsymbol{1}$	1-step
GMM1-SYS	Δ	$\Delta\boldsymbol{y}_{i,-1}, \Delta\boldsymbol{x}_i$	$\mathrm{diag}(y_{i,t-3}y_{i,t-2}), \Delta\boldsymbol{x}_i$	1-step
	levels:	$\boldsymbol{y}_{i,-1}, \boldsymbol{x}_i, \boldsymbol{1}$	$\mathrm{diag}(\Delta y_{i,t-2}), \boldsymbol{x}_i, \boldsymbol{1}$	
GMM2-SYS		as GMM1-SYS	as GMM1-SYS	2-step

When $T = 5$, for example, the instruments for the differenced equations (Z^*) and level equations (Z^+) in GMM estimation are:

$$Z_i^* = \begin{pmatrix} y_{i0} & 0 & 0 & 0 & 0 & \Delta x_{i,2} \\ 0 & y_{i0} & y_{i1} & 0 & 0 & \Delta x_{i,3} \\ 0 & 0 & 0 & y_{i1} & y_{i2} & \Delta x_{i,4} \end{pmatrix}, \quad Z_i^+ = \begin{pmatrix} \Delta y_{i1} & 0 & 0 & x_{i,2} & 1 \\ 0 & \Delta y_{i2} & 0 & x_{i,3} & 1 \\ 0 & 0 & \Delta y_{i3} & x_{i,4} & 1 \end{pmatrix}$$

Here we have reverted to counting from zero (the Ox convention), so initially, the available observations are $0 \cdots 4$. One observation is lost owing to the lagged dependent variable, and one more by differencing. The GMM intruments for the differences are specified by `Gmm("Y",2,3)`, so lagged two periods. In combined estimation, the GMM instruments for the levels are specified by `GmmLevel("Y",1,1)`, corresponding to differences, one period lagged.

The code in `absim1.ox` implements some experiments with $M = 100$ replications (taking about half a minute on a 500 Mhz Pentium III); `absim2.ox` adds the combined GMM-SYS estimators. Some results for $M = 1000$ are presented in Figure 1. These are similar to Table 1 of Arellano and Bond (1991) (but we used `Gmm("Y",2,3)` rather than `Gmm("Y",2,99)`), and Table 2 of Blundell and Bond (1998) (but with larger T, and an additional regressor).

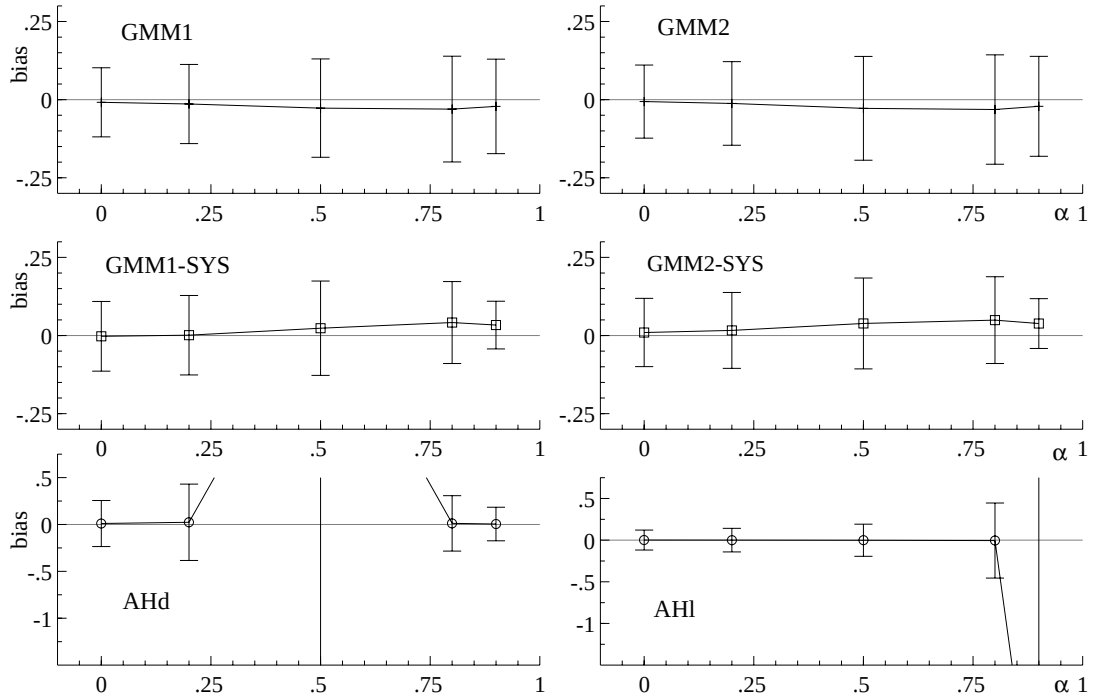


Figure 1 Mean bias of estimators, `Gmm("Y",2,3)`, $M = 1000, \beta = 1, \rho = 0.8, \phi = 0, \sigma_e^2 = 0.9$; bars are twice the MCSD.

Figure 1 shows that the GMM1 and GMM2 estimators are less precise for α close to unity. The combined estimators, however, behave much for large α. The graphs also show that AHd is unidentified at $\alpha = 0.5$, and AHl very imprecise at $\alpha = 0.9$ (see the discussion in Arellano and Bond, 1991, p.285).

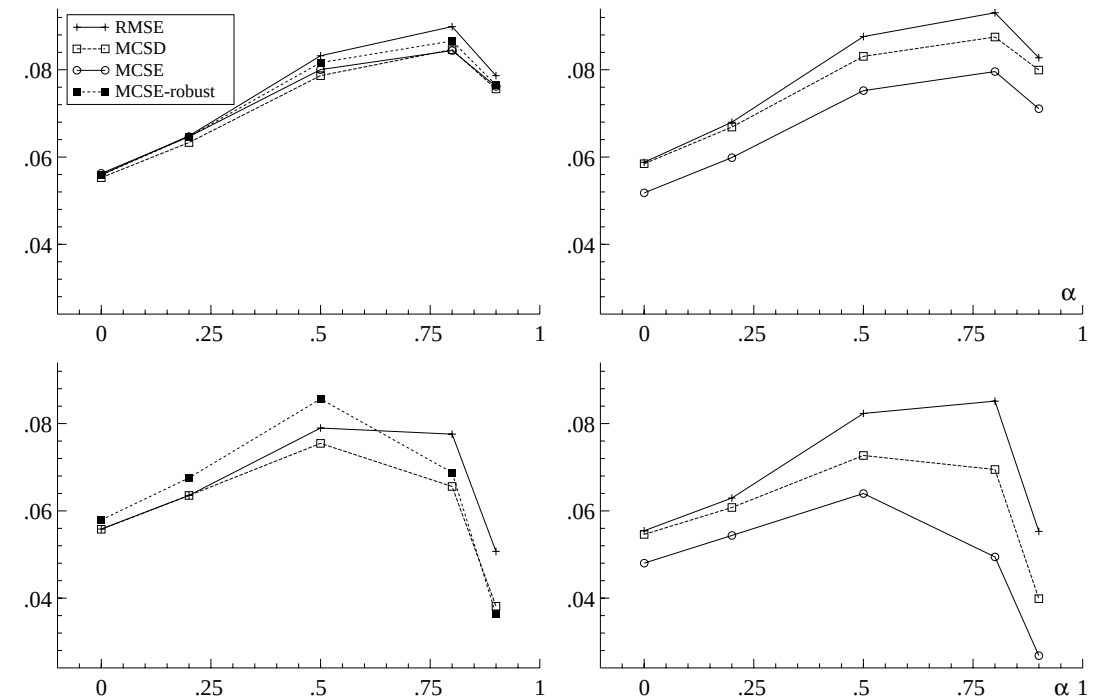


Figure 2 RMSE, MCSD and MCSE, `Gmm("Y",2,3)`, $M = 1000, \beta = 1, \rho = 0.8, \phi = 0, \sigma_e^2 = 0.9$.

Figure 2 shows that inference is problematic using the two-step standard errors. It graphs the root of the mean squared error (RMSE), the Monte Carlo standard deviation (MCSD; the standard deviation of the estimated $\hat{\alpha}$), and the the Monte Carlo standard error (MCSE; the mean of the estimated $\mathsf{SE}[\hat{\alpha}]$).

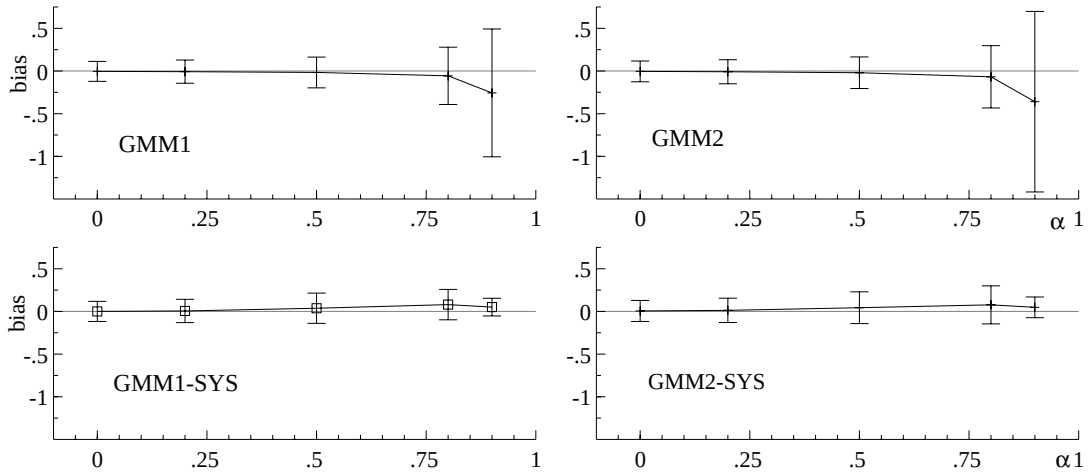


Figure 3 Mean bias of estimators, `Gmm("Y",2,2)`, $M = 1000, \beta = 1, \rho = 0.8, \phi = 0, \sigma_e^2 = 0.9$.

Figure 3 is the same experiment as 1, except using `Gmm("Y",2,2)` as instruments instead of `Gmm("Y",2,3)`. It shows that the bias for the GMM1 and GMM2 estimators is now larger for α close to unity (for example, in the first plot the bias at $\alpha = 0.9$ is -0.26, corresponding to $\bar{\hat{\alpha}} = 0.64$). In this case GMM-SYS is again an improvement, although standard errors are less reliable than in 2.

4 Static panel data estimation

4.1 Introduction

DPD/Ox can be used to compute some of the standard panel data methods, such as OLS and within-groups estimation. The following table lists the pre-programmed estimators:

	`SetTransform`	`SetMethod`	Example
OLS in levels	`T_NONE`	`M_1STEP`	`grunest1.ox`
Between estimator	`T_BETWEEN`	`M_1STEP`	`grunest1.ox`
Within estimator	`T_WITHIN`	`M_1STEP`	`grunest1.ox`
Feasible GLS	—	`M_DPD_GLS`	`grunest2.ox`
GLS (OLS residuals)	—	`M_DPD_GLS1`	`grunest2.ox`
Maximum likelihood (ML)	—	`M_DPD_ML`	`grunest2.ox`

`SetTransform` and `SetMethod` are the DPD functions which are used to select the method. Feasible generalized least squares combines within and between estimation (called SWAR in Baltagi, 1995, §2.6, after Swamy and Arora, 1972). The second GLS entry uses OLS residuals (called WALHUS in Baltagi, 1995, §2.6, after Wallace and Hussain, 1969). ML corresponds to iterated GLS.

Least squares dummy variables estimation (LSDV) can be implemented by regression on individual dummies.

4.2 Example on standard panel data estimation

This first example replicates rows one to three of Table 2.1 in Baltagi (1995), and is provided in the file `grunest1.ox`.

grunest1.ox

```
#include <oxstd.h>
#import <packages/dpd/dpd>

main()
{
    decl dpd = new DPD();  // declare and create DPD object

    dpd.Load("grunfeld.xls");                  // load data
    dpd.Info();        // print database summary statistics
    dpd.SetYear("Year");      // specify columns with years

    dpd.Select(Y_VAR, {"I", 0, 0});      // formulate model
    dpd.Select(X_VAR, {"F_1", 0, 0, "C_1", 0, 0});

    //------------------- OLS -----------------------------
    dpd.SetTransform(T_NONE);         // estimate in levels
    print("\n\n***** Baltagi (1995), Table 2.1: OLS");
    dpd.Estimate();                    // 1-step estimation

    //------------------- between -------------------------
    dpd.SetTransform(T_BETWEEN); // estimate between groups
    print("\n\n***** Baltagi (1995), Table 2.1: Between");
    dpd.Estimate();                    // 1-step estimation

    //------------------- within --------------------------
    dpd.SetTransform(T_WITHIN);   // estimate within groups
    print("\n\n***** Baltagi (1995), Table 2.1: Within");
    dpd.Estimate();                    // 1-step estimation

    delete dpd;                     // finished with object
}
```

This example lets DPD work out the individual index from the year variable, as noted in the output:

grunest1.out

```
DPD package version 1.2, object created on 20-06-2001
---- Database information ----
Sample:    1 - 200 (200 observations)
Frequency: 1
Variables: 4

Variable        #obs  #miss         min        mean         max     std.dev
Year             200      0        1935      1944.5        1954      5.7663
I                200      0        0.93      145.96      1486.7      216.33
F_1              200      0       58.12      1081.7      6241.7      1311.2
C_1              200      0         0.8      276.02      2226.3      300.35

***** Baltagi (1995), Table 2.1: OLS
DPD( 1) Modelling I by 1-step

                  ----   1-step estimation using DPD ----
                  Coefficient  Std.Error  t-value  t-prob
F_1                  0.115562   0.005836     19.8   0.000
C_1                  0.230678    0.02548     9.05   0.000
Constant            -42.7144      9.512    -4.49   0.000

sigma                 94.4084  sigma^2              8912.947
R^2                  0.812408
RSS               1755850.4841  TSS               9359943.9289
no. of observations       200  no. of parameters           3
Warning: standard errors not robust to heteroscedasticity
```

(continued)

```
Transformation used:     none

constant:                 yes  time dummies:              0
number of individuals      10 (derived from year)
longest time series        20 [1935 - 1954]
shortest time series       20 (balanced panel)

Wald (joint):    Chi^2(2) =     853.2 [0.000] **
Wald (dummy):    Chi^2(1) =     20.17 [0.000] **

***** Baltagi (1995), Table 2.1: Between
DPD( 2) Modelling I by 1-step

                  ----   1-step estimation using DPD ----
                  Coefficient  Std.Error  t-value  t-prob
F_1                  0.134646    0.02875     4.68   0.002
C_1                 0.0320315     0.1909    0.168   0.872
Constant             -8.52711      47.52   -0.179   0.863

sigma                 85.02366  sigma^2              7229.023
R^2                  0.8577682
RSS               50603.161076  TSS              355779.58273
no. of observations        10  no. of parameters           3
Warning: standard errors not robust to heteroscedasticity

Transformation used:     between groups (using individual means)

constant:                 yes  time dummies:              0
Dummies entered in transformed form
number of individuals      10 (derived from year)
longest time series        20 [1935 - 1954]
shortest time series       20 (balanced panel)

Wald (joint):    Chi^2(2) =     42.22 [0.000] **
Wald (dummy):    Chi^2(1) =   0.03221 [0.858]

***** Baltagi (1995), Table 2.1: Within
DPD( 3) Modelling I by 1-step

                  ----   1-step estimation using DPD ----
                  Coefficient  Std.Error  t-value  t-prob
F_1                  0.110124    0.01186     9.29   0.000
C_1                  0.310065    0.01735     17.9   0.000

sigma                 52.76797  sigma^2              2784.458
R^2                  0.7667576
RSS              523478.14739  TSS              2244352.2743
no. of observations       200  no. of parameters          12
Warning: standard errors not robust to heteroscedasticity

Transformation used:     within groups (deviation from individual means)

constant:                  no  time dummies:              0
number of individuals      10 (derived from year)
longest time series        20 [1935 - 1954]
shortest time series       20 (balanced panel)

Wald (joint):    Chi^2(2) =     618.0 [0.000] **
```

Further examples are:

- `grunest2.ox` which estimates the model using feasible GLS and maximum likelihood.
- `grunest3.ox` which shows how recursive estimation can be implemented.

5 DPD in OxPack for GiveWin

DPD in **OxPack** for GiveWin has an effective graphical user interface for interactive data and model selection, estimation, and testing.

5.1 Installing DPD in OxPack for GiveWin

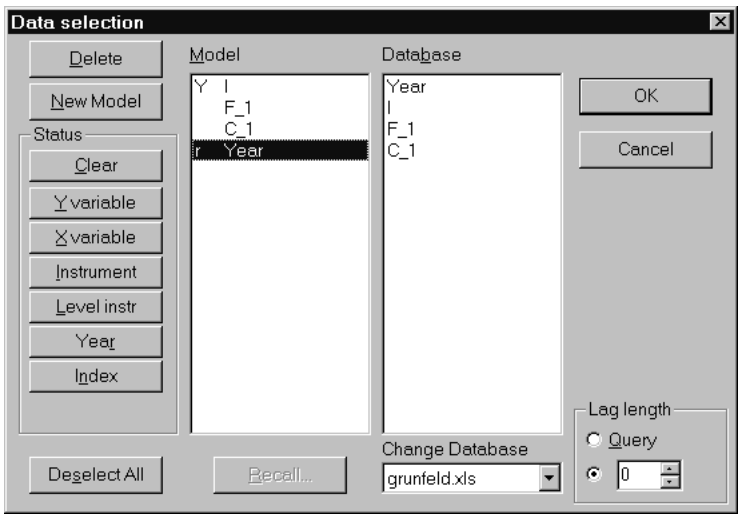
Installation of the interactive version of DPD:

(1) Install DPD into `ox/packages/dpd` as described above.
(2) Install OxPack for GiveWin (available from `http://www.nuff.ox.ac.uk/users/doornik`). OxPack requires a properly installed GiveWin version 1.20 or later. Check the version number in the GiveWin Help menu.
(3) Start GiveWin, and then OxPack from the GiveWin Modules menu. From the OxPack Package menu Choose Add/Remove Package. Locate `dpd.oxo` (in the `dpd` folder) using the Browse button, and press Add.

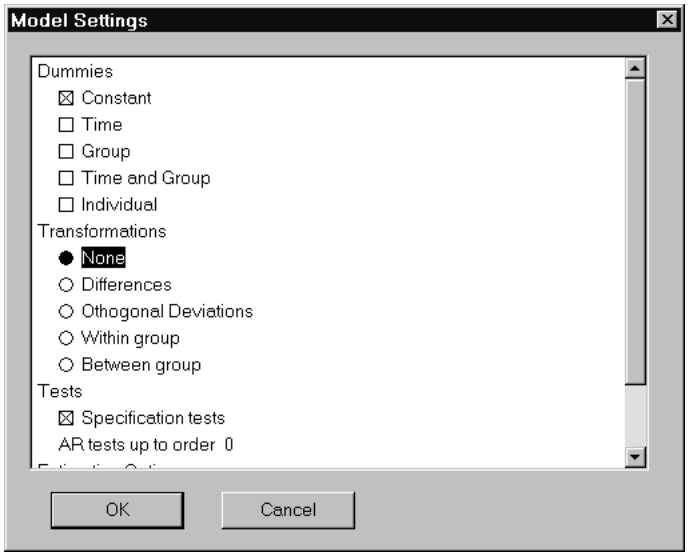
5.2 Sample session using DPD in OxPack: static panel methods

OxPack is now ready to use DPD. As a first example, we use the Grunfeld data set which is in the `dpd` folder:

(1) Load the data set `grunfeld.xls` in GiveWin.
(2) From the OxPack Package menu choose DPD. The title bar of the OxPack window shows that `DPD` is loaded and the message `DPD package version 1.00, object created on ...` is displayed in the GiveWin Results window.
(3) From the OxPack Model menu choose Formulate, and select *I*, *F_1*, and *C_1*. Add these to the model without lags. *I* will be marked with Y, indicating that it is the dependent variable (if not: select *I* in the model and click on Y-variable). Next add *Year*, select it in the model by clicking on it, and hit the Year button. The variable is now marked with an r; without a Year variable, DPD cannot estimate any models. The dialog will look like:

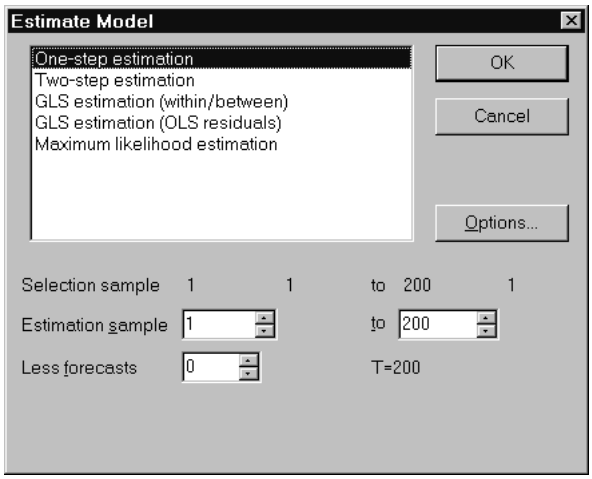


(4) Click OK, which takes you to the Functions dialog. These will be required for GMM estimation, but are not needed here, so click OK. The next dialog is used to specify the dummies and transformations, as well as some additional options. To start with OLS estimation, set Transformations to None:



Note that none of the estimation options are set.

(5) In the estimation dialog, choose one-step estimation:

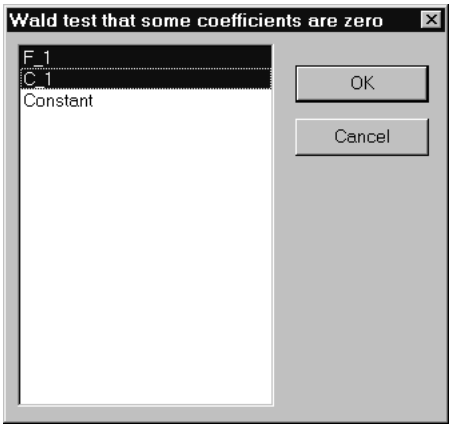


Note that the sample selection options are ignored by DPD: they do not take the panel aspect of the data into account. Section 5.4 shows how to estimate over a subsample.

(6) Estimation is nearly instantaneous, and the output appears in the GiveWin Results window. Various options are available in the Test menu. Dynamic analysis is not interesting here, as the estimated model is static. The output lists:

```
Wald (joint):    Chi^2(2) =     853.2 [0.000] **
Wald (dummy):    Chi^2(1) =     20.17 [0.000] **
```

which can be replicated easily using Test/Exclusion restrictions. For the first, select *F_1* and *C_1* for exclusion:



For the second select the Constant term.

(7) To implement the remaining static estimators, make the following choices in the Model Settings and Estimate Model dialogs:

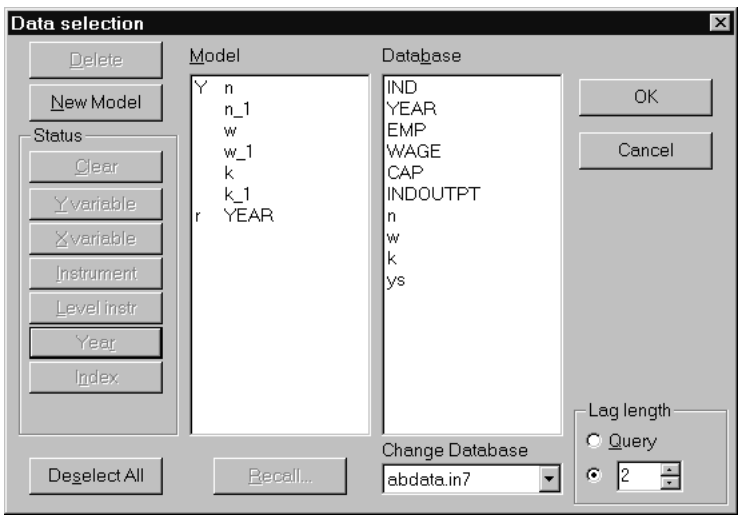
	Transformation	Estimation Method
Between estimator	Between group	One-step estimation
Within estimator	Within group	One-step estimation
Feasible GLS	None	GLS (within/between)
GLS (OLS residuals)	None	GLS (OLS residuals)
Maximum likelihood (ML)	None	Maximum likelihood estimation

For LSDV select a constant and individual dummies in Model Settings, no transformations, and estimate by one-step estimation.

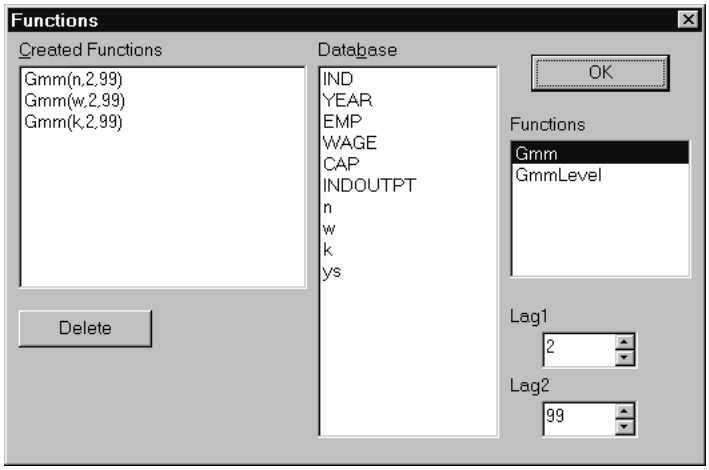
5.3 Sample session using DPD in OxPack: dynamic panel methods

The objective here is to replicate the column labelled GMM-DIF in Table 1.

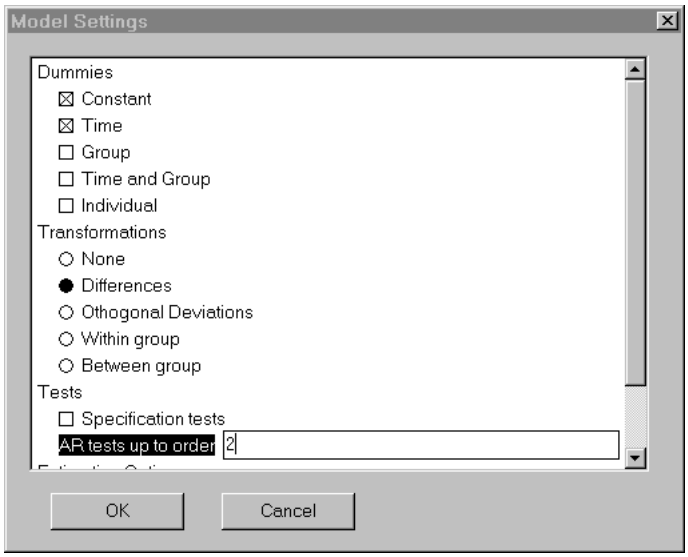
- From the OxPack Model menu choose Formulate, and select the model as shown:



- Click OK, which takes you to the Functions dialog. Three GMM functions must be formulated. Set Lag1 to 2, and Lag2 to 99; select n in the Database listbox, and click on the Add button. Repeat this for w and k:



- The next dialog is the Model Settings dialog. Select a constant and time dummies; differences for the transformation; specification tests and AR test up to order 2:

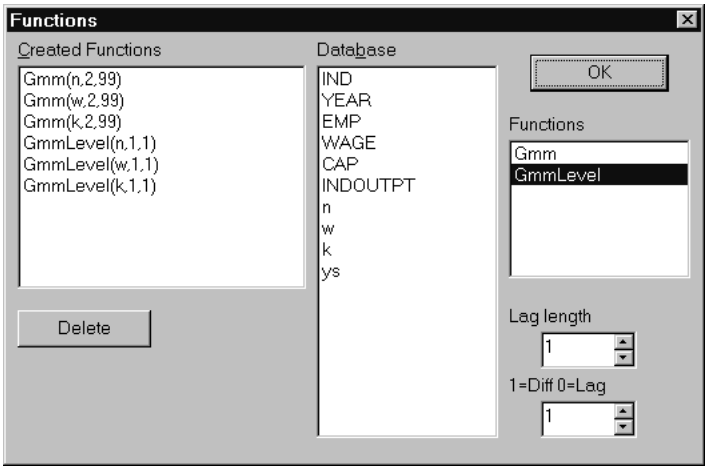


- In the estimation dialog, choose two-step estimation. The results should match those in the table.

5.4 Sample session using DPD in OxPack: combined dynamic panel methods

The next step is to replicate the last column of Table 1, labelled GMM-SYS. The basic model is unchanged from the previous section. So, select Formulate, and press OK.

The three GMM instruments for the differenced equations are unchanged, but three level instruments must be added. Select the GmmLevel function, noting that the text above the two argument edit boxes changes. Needed are the differences at lag one, so set both to one, and add the three GmmLevel instruments:



Click on OK, and select Combined estimation in the Model Settings dialog. Then choose 2-step estimation: the results should match the last column in Table 1.

Note that while OxPack is running, the program disappears from screen, and the title on the taskbar changes from 'DPD – OxPack' to 'Ox running...'. If the program takes too long, click on it on the taskbar: then press the Abort button on OxPack if you wish to interrupt it.

To estimate the first two columns of Table 1, use GiveWin to create a new year variable to use in the estimation. The GiveWin Algebra code is:

```
yearsub = YEAR < 1979 ? MISSING : YEAR;
```

Note that the sample reported in the DPD-Ox output is after allowing for lags and differences, whereas those in Table 1 include the lags.

6 Notes and remarks

6.1 Differences with the Gauss version

- DPD/Ox uses the Student-t distribution to report p-values instead of standard normal.
- The AR test is somewhat different after 2-step estimation: DPD/Gauss uses 1-step residuals in the denominator (2-step in the numerator). DPD/Ox uses 2-step residuals throughout.
- DPD/Gauss computes within-groups estimator by applying OLS after orthogonal deviations transformation. DPD/Ox implements the standard within-groups estimation by applying OLS after subtracting the individual means, as well as the orthogonal deviations method.
- DPD/Gauss has a difference between `gmm(.,2,-2)` and `gmm(.,-2,2)`. In DPD/Ox both yield identical sets of instruments.

Changes in version 1.20

- Robust standard errors for two-step GMM is now based on a small-sample correction by Windmeijer (2000). Consequently, all derived test statistics are also different.
- Robust standard errors is now the default.
- small changes to the output format.

Numerical changes in version 1.00

- Sargan test after two-step estimation is slightly different. If the current test outcome is denoted as S, the value found in the previous version of DPD was $S\hat{\sigma}_1^2/\hat{\sigma}_2^2$.

6.2 New in version 1.00

- combined differenced (or orthogonal deviations) and levels estimation (as in Blundell and Bond, 1998),
- static panel data estimation (feasible GLS, ML),
- using generalized inverses for singular moments (warning is printed),
- fairly extensive internal code revisions,
- extra export functions,
- individuals with insufficient observations are automatically removed,
- option to transform dummies or to concentrate dummies,
- option to print GMM-instrument lay-out,
- simulation class `DPDSim`,
- can be used interactively, using GiveWin and OxPack.

To do

- implement `T_TREND` and `T_SEASONALS`,
- check with quarterly data,
- some descriptive statistics.

References

Alvarez, J., and Arellano, M. (1998). The time series and cross-section asymptotics of dynamic panel data estimators. mimeo, CEMFI, Madrid.

Anderson, T. W., and Hsiao, C. (1982). Formulation and estimation of dynamic models using panel data. *Journal of Econometrics*, **18**, 47–82.

Arellano, M. (1988). An alternative transformation for fixed effects models with predetermined variables. Applied economics discussion paper, Institute of Economics and Statistics.

Arellano, M. (1995). On the testing of correlated effects with panel data. *Journal of Econometrics*, **59**, 87–97.

Arellano, M., and Bond, S. R. (1991). Some tests of specification for panel data: Monte Carlo evidence and an application to employment equations. *Review of Economic Studies*, **58**, 277–297.

Arellano, M., and Bover, O. (1995). Another look at the instrumental variables estimation of error-components models. *Journal of Econometrics*, **68**, 29–51.

Baltagi, B. H. (1995). *Econometric Analysis of Panel Data*. Chichester: John Wiley and Sons.

Blundell, R. W., and Bond, S. R. (1998). Initial conditions and moment restrictions in dynamic panel data models. *Journal of Econometrics*, **87**, 115–143.

Chamberlain, G. (1984). Panel data. in Griliches, and Intriligator (1984).

Doornik, J. A. (2001). *Object-Oriented Matrix Programming using Ox* 4th edn. London: Timberlake Consultants Press.

Griliches, Z., and Intriligator, M. D. (eds.)(1984). *Handbook of Econometrics*, Vol. 2–3. Amsterdam: North-Holland.

Hendry, D. F., and Doornik, J. A. (2001). *Empirical Econometric Modelling using PcGive: Volume I* 3rd edn. London: Timberlake Consultants Press.

Swamy, P. A. V. B., and Arora, S. S. (1972). The exact finite sample properties of the estimators of coefficients in the error component regressions models. *Econometrica*, **40**, 261–275.

Wallace, T. D., and Hussain, A. (1969). The use of error components models in combining cross-section and time-series data. *Econometrica*, **37**, 55–72.

White, H. (1982). Instrumental variables regression with independent observations. *Econometrica*, **50**, 483–499.

Windmeijer, F. (2000). A finite sample correction for the variance of linear two-step GMM estimators. IFS working paper W00/19 (www.ifs.org.uk), London: The Institute for Fiscal Studies.

A Technical Appendix

This section gives a summary of the statistical output of DPD, giving the formulae which are used in the computations. The notation uses Ox-style indexing which starts from 0.

A.1 Transformations

- none
$$x^*_{it} = x_{it}, \quad t = 0, \ldots, T_i - 1.$$
- first differences
$$\begin{aligned} x^*_{it} = \Delta x_{it} &= x_{it} - x_{i,t-1}, \quad t = 1, \ldots, T_i - 1, \\ x^*_{i0} = \Delta x_{i0} &= 0. \end{aligned}$$
- time means:
$$\bar{x}_i = \frac{1}{T_i} \sum_{s=0}^{T_i-1} x_{is}.$$
- deviations from time means
$$x^*_{it} = x_{it} - \bar{x}_i, \quad t = 0, \ldots, T - 1.$$
- orthogonal deviations
$$x^o_{it} = \left(x_{it} - \frac{1}{T_i - t} \sum_{s=t+1}^{T_i-1} x_{is} \right) \left(\frac{T_i - t}{T_i - t + 1} \right)^{1/2}, \quad t = 1, \ldots, T_i - 2.$$
The orthogonal deviations are stored with one lag, so that the first observation is lost instead of the last (this brings it in line with first differencing):
$$\begin{aligned} x^*_{it} &= x^o_{i,t-1}, \quad t = 1, \ldots, T_i - 1. \\ x^*_{i0} &= 0. \end{aligned}$$
- GLS deviations:
$$x^*_{it} = x_{it} - \theta_i \bar{x}_i, \quad t = 0, \ldots, T_i - 1.$$
Where the choice of θ_i determines the method, as discussed in the next section.

A.2 Static panel-data estimation

The static single-equation panel model can be written as:

$$y_{it} = \boldsymbol{x}'_{it}\boldsymbol{\gamma} + \lambda_t + \eta_i + v_{it}, \quad t = 0, \ldots, T - 1, \quad i = 0, \ldots, N - 1.$$

The λ_t and η_i are respectively time and individual specific effects and $\boldsymbol{x}_{it}$ is a k^* vector of explanatory variables. N is the number of cross-section observations. The total number of observations is $O = NT$.

In an unbalanced panel, different individuals may have a different number of observations, T_i. In that case the total number of observations equals $O = \sum T_i$.

Stacking the data for an individual according to time:

$$\boldsymbol{y}_i = \boldsymbol{X}_i\boldsymbol{\gamma} + \boldsymbol{\lambda} + \boldsymbol{\iota}_i\eta_i + \boldsymbol{v}_i, \quad i = 0, \ldots, N - 1,$$

where $\boldsymbol{\iota}_i$ is a column of ones. Using $\mathbf{D}$ for the matrix with dummies:

$$\boldsymbol{y}_i = \boldsymbol{X}_i\boldsymbol{\gamma} + \boldsymbol{D}_i\boldsymbol{\delta} + \boldsymbol{v}_i, \quad i = 0, \ldots, N - 1.$$

The next step is to stack all individuals, combining the xs and dummies into $\boldsymbol{W} = [\boldsymbol{X} : \boldsymbol{D}]$:

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{\beta} + \boldsymbol{v}.$$

Baltagi (1995) reviews the standard estimation methods used in this setting.

- The *OLS* estimates are:

$$\begin{aligned}\hat{\boldsymbol{\beta}} &= (\boldsymbol{W}'\boldsymbol{W})^{-1}\boldsymbol{W}'\boldsymbol{y},\\ \widehat{\mathbb{V}}\left[\hat{\beta}\right] &= \hat{\sigma}_v^2(\boldsymbol{W}'\boldsymbol{W})^{-1},\\ \hat{\sigma}_v^2 &= \hat{\boldsymbol{v}}'\hat{\boldsymbol{v}}/(n-p),\\ \hat{\boldsymbol{v}} &= \boldsymbol{y}-\boldsymbol{W}\hat{\boldsymbol{\beta}}.\end{aligned}$$

  Here $\boldsymbol{W}$ has k columns, containing all specified regressors including dummies, sp $p = k$.
- The *LSDV* (least squares dummy variables) estimates use individual dummies.
- The *within* estimates replace $\boldsymbol{y}$ and $\boldsymbol{X}$ by deviations from time means.
- The *between* estimates replace $\boldsymbol{y}$ and $\boldsymbol{W}$ by the individual means.
- The *feasible GLS* estimates replace $\boldsymbol{y}$ and $\boldsymbol{W}$ by deviations from weighted time means. The outcome depends on the choice of theta, which in DPD is set by specifying σ_v and σ_η^2:

$$\theta_i = 1 - \frac{\sigma_v}{\sigma_i}, \quad \sigma_i^2 = \sigma_v^2 + T_i\sigma_\eta^2.$$

  When OLS residuals $\boldsymbol{u} = \hat{\boldsymbol{v}}_{OLS}$ are used, the GLS estimator can be based on:

$$\hat{\sigma}_v^2 = \frac{\sum_{i=0}^{N-1}\sum_{t=0}^{T_i-1}(u_{it}-\bar{u}_i)^2}{O-N},$$

$$\hat{\sigma}_i^2 = \sigma_0^2 = \frac{\sum_{i=0}^{N-1} T_i\bar{u}_i}{N-1}.$$

  The Ox code computes θ_i from σ_v and σ_η^2 with the latter derived from $\hat{\sigma}_0^2$ as:

$$\hat{\sigma}_\eta^2 = \left(\hat{\sigma}_0^2 - \hat{\sigma}_v^2\right)\frac{N}{O}.$$

  In a balanced panel: $N/O = 1/T$. This is not optimal in an unbalanced panel, but seems reasonable.
  The standard feasible GLS estimator uses the between and within estimates:

$$\begin{aligned}\sigma_v^2 &= \hat{\sigma}^2_{\textit{within}},\\ \sigma_i^2 &= T_i\hat{\sigma}^2_{\textit{between}}.\end{aligned}$$

  For convenience, θ_i is specified using three variance components σ_v^2, σ_a^2, and σ_η^2:

$$\theta_i = 1 - \left(\frac{\sigma_v^2}{\sigma_a^2 + T_i\sigma_\eta^2}\right)^{1/2}. \tag{2}$$

	σ_v^2	σ_a^2	σ_η^2
OLS based GLS	$\hat{\sigma}_v^2$	$\hat{\sigma}_v^2$	$\hat{\sigma}_\eta^2$
between/within based GLS	$\hat{\sigma}^2_{\textit{within}}$	0	$\hat{\sigma}^2_{\textit{between}}$

- The *maximum likelihood* estimates obtain θ by iterating the GLS procedure. The concentrated likelihood is:

$$\ell_c(\tau;\hat{\boldsymbol{\beta}}(\tau),\hat{\sigma}_v^2(\tau))/O = c - 0.5\log(\hat{\sigma}_v^2) - 0.5\sum_{i=0}^{N-1}\log(1+T_i\tau)/O, \tag{3}$$

  where $\tau = \sigma_\eta^2/\sigma_v^2$, so that:

$$\theta_i = 1 - (1+T_i\tau)^{-1/2}.$$

To summarize the implementation in DPD:

	number of observations, n	degrees of freedom lost in estimation, p	transforms dummies
OLS (no transformation)	O	k	no
within	O	$k+N$	no
between	N	k	yes
GLS, ML	O	k	yes

It is important to note that the within transformations are only applied to $\boldsymbol{X}$, and not to the dummies **D**. In the within estimator, individual dummies are redundant after subtracting the individual means. In the between estimator, time dummies are irrelevant, and individual dummies create a perfect fit. In GLS and ML the individual means are subtracted with weights.

A.3 Dynamic panel data estimation

The single equation dynamic panel model can be written as:

$$y_{it} = \sum_{s=1}^{m} \alpha_i y_{i,t-s} + \boldsymbol{x}_{it}'\boldsymbol{\gamma} + \lambda_t + \eta_i + v_{it}, \quad t = 0, \ldots, T_i - 1, \quad i = 0, \ldots, N-1.$$

As before, λ_t and η_i are time and individual specific effects and $\boldsymbol{x}_{it}$ is a k^* vector of explanatory variables. It is assumed that allowance for lagged observations has been made, so observations on $y_{i,-s}, \ldots, y_{i,-1}$ are available. The total number of observations available for estimation equals $O = \sum T_i$.

Stacking the data for an individual according to time: and using **D** for the matrix with dummies:

$$\boldsymbol{y}_i = \boldsymbol{X}_i\boldsymbol{\gamma} + \boldsymbol{D}_i\boldsymbol{\delta} + \boldsymbol{v}_i, \quad i = 0, \ldots, N-1.$$

The estimators and tests are described in Arellano and Bond (1991), and have the form:

$$\begin{array}{lcl} \boldsymbol{M} & = & \left(\sum_i \boldsymbol{W}_i'\boldsymbol{Z}_i\right)\boldsymbol{A}_N\left(\sum_i \boldsymbol{Z}_i'\boldsymbol{W}_i\right), \\ \boldsymbol{A}_N & = & \left(\sum_i \boldsymbol{Z}_i'\boldsymbol{H}_i\boldsymbol{Z}_i\right)^{-1}, \\ \hat{\boldsymbol{\beta}} & = & \boldsymbol{M}^{-1}\left(\sum_i \boldsymbol{W}_i'\boldsymbol{Z}_i\right)\boldsymbol{A}_N\left(\sum_i \boldsymbol{Z}_i'\boldsymbol{q}_i\right), \\ \hat{\sigma}_u^2 & = & \hat{\boldsymbol{u}}'\hat{\boldsymbol{u}}/(n-p), \\ \hat{\boldsymbol{u}} & = & \boldsymbol{q} - \boldsymbol{W}\hat{\boldsymbol{\beta}}. \end{array} \tag{4}$$

In formulating the model, we distinguish the following types of variables:

$\boldsymbol{y}$	dependent variables,
$\boldsymbol{X}$	regressors, including lagged dependent variables,
$\boldsymbol{D}$	dummy variables,
$\boldsymbol{I}$	normal instruments,
$\boldsymbol{L}$	'level' instruments,
$\boldsymbol{G}$	GMM-style instruments.

The $\boldsymbol{G}_i$ are described under the `Gmm` function. From these variables, the dependent variable, regressor and instrument matrices are formed as follows:

			dimension
$\boldsymbol{q}_i$	=	$\boldsymbol{y}_i^*$	$T_i \times 1$
$\boldsymbol{W}_i$	=	$[\boldsymbol{X}_i^* : \boldsymbol{D}_i^s]$	$T_i \times k$
$\boldsymbol{Z}_i$	=	$[\boldsymbol{G}_i : \boldsymbol{I}_i^* : \boldsymbol{D}_i^s : \boldsymbol{L}_i]$	$T_i \times z$

Where the * denotes some transformation which can be chosen from §A.1. This also affects the degrees of freedom:

transformation	number of observations, n	degrees of freedom lost in estimation, p	transforms dummies
differences	$O-N$	k	$\boldsymbol{D}^s=\boldsymbol{D}$ (optional: $\boldsymbol{D}^*$)
deviations	$O-N$	k	$\boldsymbol{D}^s=\boldsymbol{D}$ (optional: $\boldsymbol{D}^*$)
none	O	k	no: $\boldsymbol{D}^s=\boldsymbol{D}$
within	O	$k+N$	no: $\boldsymbol{D}^s=\boldsymbol{D}$
between	N	k	yes: $\boldsymbol{D}^s=\boldsymbol{D}^*$
GLS	O	k	yes: $\boldsymbol{D}^s=\boldsymbol{D}^*$

When the option to **concentrate** out dummies is used, $\boldsymbol{y}^*$, $\boldsymbol{X}^*$, $\boldsymbol{I}^*$, and $\boldsymbol{L}$ are replaced by the residuals from regressing on the set of dummies $\boldsymbol{D}^s$. Subsequently, the dummies are omitted from further computations.

In **one-step estimation**, $\boldsymbol{H}_i$ is the identity matrix, except when estimating in first differences:

$$\boldsymbol{H}_{1,i}=\boldsymbol{I}_{T_i},\ \text{except: } \boldsymbol{H}_{1,i}^{\text{diff}}=\begin{pmatrix} 1 & -1/2 & 0 & \cdots & 0\\ -1/2 & 1 & -1/2 & & 0\\ 0 & -1/2 & 1 & & 0\\ \vdots & & & & \vdots\\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}. \tag{5}$$

In **two-step estimation** $\boldsymbol{H}_i$ is based on the previous step residuals:

	$\boldsymbol{H}_i$	$\boldsymbol{M}$	$\boldsymbol{A}_N$	$\hat{\boldsymbol{u}}$
one-step	$\boldsymbol{H}_{1,i}$, see (5)	$\boldsymbol{M}_1$	$\boldsymbol{A}_{1,N}$	$\hat{\boldsymbol{u}}_{1,i}=\hat{\boldsymbol{v}}_{1,i}^*$
two step	$\boldsymbol{H}_{2,i}=\hat{\boldsymbol{v}}_{1,i}^*\hat{\boldsymbol{v}}_{1,i}^{*\prime}$	$\boldsymbol{M}_2$	$\boldsymbol{A}_{2,N}$	$\hat{\boldsymbol{u}}_{2,i}=\hat{\boldsymbol{v}}_{2,i}^*$

with a subscript added to distinguish between one and two step estimation.

The one-step estimated **variance** of the coefficients is:

$$\widehat{\mathsf{V}}_1\left[\hat{\beta}\right]=\hat{\sigma}_{1,u}^2\boldsymbol{M}_1^{-1}.$$

If selected, robust standard errors after one-step estimation are computed as:

$$\widehat{\mathsf{V}}_{1r}\left[\hat{\beta}\right]=\boldsymbol{M}_1^{-1}\left(\sum_i\boldsymbol{W}_i'\boldsymbol{Z}_i\right)\boldsymbol{A}_{1,N}\boldsymbol{A}_{2,N}^{-1}\boldsymbol{A}_{1,N}\left(\sum_i\boldsymbol{Z}_i'\boldsymbol{W}_i\right)\boldsymbol{M}_1^{-1}.$$

The two-step asymptotic variance matrix is:

$$\widehat{\mathsf{V}}_2\left[\widehat{\beta}\right]=\boldsymbol{M}_2^{-1}.$$

This variance estimator can be severely downward biased in small samples, and is therefore only reported when robust standard errors is switched off. The preffered solution (and the default) is to use the robust variance for two-step estimation, $\widehat{\mathsf{V}}_{2r}[\widehat{\beta}]$, which uses the small sample correction as derived by Windmeijer (2000).

The **AR test** for order m is computed as:

$$\mathsf{AR}(m)=\frac{d_0}{(d_1+d_2+d_3)^{1/2}}. \tag{6}$$

Using $\boldsymbol{w}_i$ for the residuals lagged m periods (substituting zero for missing lags):

$$w_{it}=u_{i,t-m}\text{ for }t=m,\ldots,T_i,\text{ and }\quad w_{it}=0\text{ for }t<m.$$

The d_i are defined as:

$$\begin{aligned} d_0 &= \textstyle\sum_i \boldsymbol{w}_i' \boldsymbol{u}_i, \\ d_1 &= \textstyle\sum_i \boldsymbol{w}_i' \boldsymbol{H}_i \boldsymbol{w}_i, \\ d_2 &= -2 \left(\textstyle\sum_i \boldsymbol{w}_i' \boldsymbol{W}_i\right) \boldsymbol{M}^{-1} \left(\textstyle\sum_i \boldsymbol{W}_i' \boldsymbol{Z}_i\right) \boldsymbol{A}_N \left(\textstyle\sum_i \boldsymbol{Z}_i' \boldsymbol{H}_i \boldsymbol{w}_i\right), \\ d_3 &= \left(\textstyle\sum_i \boldsymbol{w}_i' \boldsymbol{W}_i\right) \widehat{\mathrm{V}} \left[\hat{\beta}\right] \left(\textstyle\sum_i \boldsymbol{W}_i' \boldsymbol{w}_i\right). \end{aligned} \tag{7}$$

The components are used as the notation suggests, except for $\boldsymbol{H}_i$ in one-step estimation:[7]

	$\boldsymbol{H}_i$	$\boldsymbol{M}$	$\boldsymbol{A}_N$	$\boldsymbol{u}_i$	$\widehat{\mathrm{V}}[\hat{\beta}]$
one step:	$\hat{\sigma}^2_{1,u} \boldsymbol{H}_{1,i}$	$\boldsymbol{M}_1$	$\boldsymbol{A}_{1,N}$	$\hat{\boldsymbol{v}}^*_{1,i}$	$\widehat{\mathrm{V}}_1[\hat{\beta}]$
one step, robust:	$\boldsymbol{H}_{2,i}$	$\boldsymbol{M}_1$	$\boldsymbol{A}_{1,N}$	$\hat{\boldsymbol{v}}^*_{1,i}$	$\widehat{\mathrm{V}}_{1r}[\hat{\beta}]$
two step:	$\boldsymbol{H}_{2,i}$	$\boldsymbol{M}_2$	$\boldsymbol{A}_{2,N}$	$\hat{\boldsymbol{v}}^*_{2,i}$	$\widehat{\mathrm{V}}_2[\hat{\beta}]$

After estimation in orthogonal deviations, the AR test (6) is based on the residuals from the differenced model as follows (the superscript Δ refers to the regressors and residuals from the model in differenced form):

$$\begin{aligned} d_0 &= \textstyle\sum_i \boldsymbol{w}_i^{\Delta\prime} \boldsymbol{u}_i^{\Delta}, \\ d_1 &= \textstyle\sum_i \boldsymbol{w}_i^{\Delta\prime} \boldsymbol{H}_i^{\Delta} \boldsymbol{w}_i^{\Delta}, \\ d_2 &= -2 \left(\textstyle\sum_i \boldsymbol{w}_i^{\Delta\prime} \boldsymbol{W}_i^{\Delta}\right) \boldsymbol{M}^{-1} \left(\textstyle\sum_i \boldsymbol{W}_i' \boldsymbol{Z}_i\right) \boldsymbol{A}_N \left(\textstyle\sum_i \boldsymbol{Z}_i' \boldsymbol{\Psi}_i\right), \\ d_3 &= \left(\textstyle\sum_i \boldsymbol{w}_i^{\Delta\prime} \boldsymbol{W}_i^{\Delta}\right) \widehat{\mathrm{V}} \left[\hat{\beta}\right] \left(\textstyle\sum_i \boldsymbol{W}_i^{\Delta\prime} \boldsymbol{w}_i^{\Delta}\right). \end{aligned} \tag{8}$$

$\boldsymbol{H}_i^{\Delta}$ is $\boldsymbol{H}_{1,i}^{\text{diff}}$ after one-step estimation, and $\boldsymbol{u}_i^{\Delta\prime} \boldsymbol{u}_i^{\Delta}$ otherwise. $\boldsymbol{\Psi}_i$ equals

$$\begin{pmatrix} \sqrt{(T_i+1)/T_i} & 0 & 0 & \cdots & 0 \\ \sqrt{(T_i-1)/(T_i-2)} & \sqrt{T_i/(T_i-1)} & 0 & \cdots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & 0 & \sqrt{1/2} & \sqrt{2/1} \end{pmatrix} \boldsymbol{w}_i$$

after one-step estimation, and $\boldsymbol{u}_i \boldsymbol{u}_i^{\Delta\prime} \boldsymbol{w}_i^{\Delta}$ after one-step robust or two-step estimation.

The **Sargan test** with $z-k$ degrees of freedom is after one-step estimation:

$$S_1 = \left(\sum_i \hat{\boldsymbol{v}}^{*\prime}_{1,i} \boldsymbol{Z}_i\right) \boldsymbol{A}_{1,N} \left(\sum_i \boldsymbol{Z}_i' \hat{\boldsymbol{v}}^*_{1,i}\right) \hat{\sigma}^{-2}_{1,u},$$

and after two-step estimation (see 1 and equation (10) in Arellano and Bond, 1991):

$$S = \left(\sum_i \hat{\boldsymbol{v}}^{*\prime}_{2,i} \boldsymbol{Z}_i\right) \boldsymbol{A}_{2,N} \left(\sum_i \boldsymbol{Z}_i' \hat{\boldsymbol{v}}^*_{2,i}\right).$$

Three **Wald tests** are routinely reported to test the significance of groups of variables:

Wald (joint): on **W**, all regressor except dummies,

Wald (dummy): on **D**, all dummies (including constant),

Wald (time): on time dummies (including the constant in the differenced/deviations model).

[7] After two-step estimation, DPD for Gauss computes the AR test using one-step residuals in the d_i in the denominator for $\sum_i \boldsymbol{w}_i' \boldsymbol{H}_i \boldsymbol{w}_i$, $\sum_i \boldsymbol{w}_i' \boldsymbol{W}_i$, and $\sum_i \boldsymbol{Z}_i' \boldsymbol{H}_i \boldsymbol{w}_i$.

A.4 Dynamic panel data, combined estimation

In combined estimation, GMM-SYS, the levels equation is used jointly with the transformed equation, see Blundell and Bond (1998). The IV estimation of (4) still applies, but the data are organized as follows:

									dimension
$\boldsymbol{q}_i$	$=$	$\begin{bmatrix} \boldsymbol{q}_i^* \\ \boldsymbol{q}_i^+ \end{bmatrix}$	$=$	$\begin{bmatrix} \boldsymbol{y}_i^* \\ \boldsymbol{y}_i \end{bmatrix}$					$2T_i \times 1$
$\boldsymbol{W}_i$	$=$	$\begin{bmatrix} \boldsymbol{W}_i^* \\ \boldsymbol{W}_i \end{bmatrix}$	$=$	$\begin{bmatrix} \boldsymbol{X}_i^* & \boldsymbol{D}_i^* \\ \boldsymbol{X}_i & \boldsymbol{D}_i \end{bmatrix}$					$2T_i \times k$
$\boldsymbol{Z}_i$	$=$	$\begin{bmatrix} \boldsymbol{Z}_i^* \\ \boldsymbol{Z}_i^+ \end{bmatrix}$	$=$	$\begin{bmatrix} \boldsymbol{G}_i & 0 & \boldsymbol{I}_i^* & 0 & \boldsymbol{L}_i \\ 0 & \boldsymbol{G}_i^+ & \boldsymbol{I}_i & \boldsymbol{D}_i & 0 \end{bmatrix}$					$2T_i \times (z + z^+)$

$\boldsymbol{G}_i^+$ are the GMM-style instruments for the levels equation, described under the `GmmLevel` function.[8] The dummies in the transformed equation are always transformed. When using differences or deviations, the effective number of observations in the transformed equations is as before ($O - N$), whereas the levels equations have O observations. Internally, both the transformed and levels equation i has T_i equations, but, when using differences/deviations, the first observation in the transformed equation is set to zero, resulting in $T_i - 1$ effective observations.

When the option to **concentrate** out dummies is used, $\boldsymbol{y}^*$, $\boldsymbol{X}^*$, $\boldsymbol{I}^*$, and $\boldsymbol{L}$ are replaced by the residuals from regressing on the set of dummies $\boldsymbol{D}^*$, and $\boldsymbol{y}$ and $\boldsymbol{X}$ on $\boldsymbol{D}$. Note that there is one time dummy extra in combined estimation compared to normal estimation when using differences or deviations.

After estimation, there are two sets of residuals:

$$\begin{aligned} \hat{\boldsymbol{u}}^* &= \boldsymbol{q}^* - \boldsymbol{W}^* \hat{\boldsymbol{\beta}} \quad \text{(transformed equation)}, \\ \hat{\boldsymbol{u}}^+ &= \boldsymbol{q}^+ - \boldsymbol{W}^+ \hat{\boldsymbol{\beta}} \quad \text{(levels equation)}. \end{aligned}$$

The reported residual variance is based on the levels residuals: $\hat{\sigma}^2_{u^+} = \hat{\boldsymbol{u}}^{+\prime}\hat{\boldsymbol{u}}^+/(n-p)$. Here $n = O = \sum_{i=0}^{N-1} T_i$ and p are as before.

In **one-step estimation** $\boldsymbol{H}_i$ is as in (5) in the transformed equations, and the identity matrix in the level equations ($1/2\boldsymbol{I}$ when using differences).[9] Only robust standard errors are reported after one-step estimation.

In **two-step estimation** $\boldsymbol{H}_i$ is based on the previous step residuals $\hat{\boldsymbol{u}}_i' = (\hat{\boldsymbol{u}}_i^{*\prime} : \hat{\boldsymbol{u}}_i^{+\prime})$.

The AR test (6) is based on the residuals from the transformed equation:

$$\begin{aligned} d_0 &= \textstyle\sum_i \boldsymbol{w}_i^{*\prime} \boldsymbol{u}_i^*, \\ d_1 &= \textstyle\sum_i \boldsymbol{w}_i^{*\prime} \boldsymbol{u}_i^* \boldsymbol{u}_i^{*\prime} \boldsymbol{w}_i^*, \\ d_2 &= -2 \left(\textstyle\sum_i \boldsymbol{w}_i^{*\prime} \boldsymbol{W}_i^*\right) \boldsymbol{M}^{-1} \left(\textstyle\sum_i \boldsymbol{W}_i' \boldsymbol{Z}_i\right) \boldsymbol{A}_N \left(\textstyle\sum_i \boldsymbol{Z}_i' \boldsymbol{u}_i \boldsymbol{u}_i^{*\prime} \boldsymbol{w}_i^*\right), \\ d_3 &= \left(\textstyle\sum_i \boldsymbol{w}_i^{*\prime} \boldsymbol{W}_i^*\right) \widehat{\mathbb{V}} \left[\hat{\beta}\right] \left(\textstyle\sum_i \boldsymbol{W}_i^{*\prime} \boldsymbol{w}_i^*\right). \end{aligned}$$

After orthogonal deviations, the * is replaced by $^\Delta$ as in (8).

[8] The code used to estimate Blundell and Bond (1998) had $\boldsymbol{Z}_i^* = \boldsymbol{G}_i \; 0 \; \boldsymbol{I}_i^* \; \boldsymbol{D}_i^* \; \boldsymbol{L}_i$.

[9] The code used to estimate in Blundell and Bond (1998) had $\boldsymbol{H}_{1,i} = \boldsymbol{I}_{T_i}$ also for estimation in differences.

B DPD exported member functions

The DPD class derives from Modelbase, which in turn derives from Database. Some of the functions below are in the base class or override a base-class virtual function but documented because they will be commonly used when estimating model. Consult the next section, and the header file dpd.h for definitions of member variables, and undocumented functions, such as those for communication with OxPack.

Notation:

g	number of GMM-type instrumental variables,
k	total number of regressors (including dummies),
m	number of normal instruments,
p	number of instrument columns (normal plus GMM-style),
N	number of cross-section observations, N,
O	total number of observations: $O = \sum T_i$,
T	maximum number of time-series observations, $T = max\{T_i\}$,
$\boldsymbol{D}$	dummies,
$\boldsymbol{I}$	normal instruments, $O \times m$,
$\boldsymbol{W}$	regressor matrix,
$\boldsymbol{Y}$	dependent variable, $O \times 1$,
Y_i	dependent variable for individual i, $T_i \times m$,
Z_i	complete instrument matrix for individual i, $T_i \times p$,
T_{db}	total number of observations in the database.

DPD::ClearModel

```
ClearModel();
```

No return value.

Description

Clears the estimated model. This will result in re-initialization of the data, so can be used e.g. to switch from orthogonal deviations to first differences, or when new data has been generated for a Monte Carlo replication.

DPD::DeSelect

```
DeSelect();
```

No return value.

Description

Clears the model formulation, i.e. clears previous calls to Select(), Gmm(), GmmLevel(), SetSelSample(), and ForceSelSample(). The remainder of the model formulation is not affected (such as the transformation and dummies).

DPD::Diff

```
Diff(const mX, const iLag);
```

mX	in:	$O \times k$ matrix, where O is the database size.
iLag	in:	lag length of difference.

Return value

Returns an $O \times k$ matrix with the panel difference. Observations which are lost are replaced by the missing value (NaN).

Description

The panel difference can only be taken if the index is known, see SetIndex() and SetYear(). The method of computing the difference is discussed in §2.1.

Note that a model is formulated in levels and then estimated in differences by specifying the transformation in `SetTransform`.

DPD::DPD

```
DPD();
```

No return value.

Description

Constructor function.

DPD::Estimate

```
Estimate();
```

Return value

Returns an `TRUE` if succesful, `FALSE` otherwise.

Description

Constructs the data matrix for estimation (by calling `InitData`, if not already done), and estimates the model. Prints the results, unless this is switched off by `SetPrint`. Use `Select` prior to `Estimate` to formulate the model.

When requesting 2-step estimation, DPD will first do 1-step estimation with robust standard errors.

DPD::Get...

Return value

`GetCovar`	estimated coefficient variance matrix, $k \times k$
`GetDummies`	returns current dummy selection, integer
`GetDummyNames`	get names of dummies, k_d array
`GetGlsSigma2`	returns σ_v^2, σ_v^2 and σ_η^2, 1×3 matrix
`GetIndex`	variable with the individual index, same no of rows as database
`GetModelStatus`	returns `M_...` model status
`GetPar`	estimated coefficients, $k \times 1$
`GetParNames`	coefficients names, including dummies, k array
`GetR2`	returns R squared, double
`GetRegNames(iVar)`	returns array names of regressors of type `iVar`
`GetResiduals`	residuals of the (transformed) equations, $T_{db} \times 1$
`GetResidualsLevels`	residuals of the levels equations, $T_{db} \times 1$
`GetResidualsDiff`	deviations: in differenced form, else as `GetResiduals`, $T_{db} \times 1$
`GetResVar`	estimated residual variance, $\hat{\sigma}^2$
`GetRSS`	residual sum of squares (for levels after combined estimation)
`GetSigma`	estimated equation standard error, $\hat{\sigma}$
`GetTest`	returns current test options (as set by `SetTest`)
`GetTestAR`	returns highest order AR test (2×1 matrix: test; p-value)
`GetTestSargan`	returns Sargan test outcome (2×1 matrix: test; p-value)
`GetTransform`	returns current transformation
`GetTSS`	total sum of squares

Description

Most of these functions can be only called after the data has been loaded for estimation, or after successful estimation.

DPD::Gmm

`Gmm(const sX, const iLag, const iLag1, const iLag2);`

`sX`	in:	string, name of variable for GMM-type instrument.
`iLag1`	in:	m_1, where $t - m_1$ is most recent observation used as instrument.
`iLag2`	in:	m_2, where $t - m_2$ is most distant observation used as instrument.

No return value.[10]

Description

Use this function to specify GMM-type instruments (in contrast to normal instruments, which are the variables specified using `I_VAR` in `Select`). The levels of the named variables are used. The instruments are dated $t - m_2 \cdots t - m_1$, so that the most recent observation is $t - m_1$ and the most distant $t - m_2$. If $m_2 < m_1$, the arguments are swapped. The instruments are automatically adjusted for time periods which consist entirely of missing data (making a DPD/GAUSS style `gmmb` unnecessary). A call to `DeSelect` will also clear all instruments.

As an example, consider a panel with two individuals, one with data from 1976 to 1981, and the second starting a year later. Say we use two lags in the model, and estimate using first differences (so another observation is lost, and we have three observations for estimation: 1979–1981). Then (remember that indexing in Ox starts at 0):

	individual 0				individual 1			
	$y_{0,t}$	$y_{0,t-2}$	$\Delta y_{0,t-2}$	s	$y_{0,t}$	$y_{0,t-2}$	$\Delta y_{0,t-2}$	s
1976	$y_{0,76}$	.	.		.	.	.	
1977	$y_{0,77}$	.	.		$y_{1,77}$	.	.	
1978	$y_{0,78}$	$y_{0,76}$	.		$y_{1,78}$	.	.	
1979	$y_{0,79}$	$y_{0,77}$	$\Delta y_{0,77}$	0	$y_{1,79}$	$y_{1,77}$	.	-1
1980	$y_{0,80}$	$y_{0,78}$	$\Delta y_{0,78}$	1	$y_{1,80}$	$y_{1,78}$	$\Delta y_{1,79}$	0
1981	$y_{0,81}$	$y_{0,79}$	$\Delta y_{0,79}$	2	$y_{1,81}$	$y_{1,79}$	$\Delta y_{1,79}$	1

The standard case, where the second lag of the dependent variable is used as the GMM-type instrument is written as :

$$Z_i = [\text{diag}(y_{i,1} \cdots y_{i,t-2})], \quad t = 3, \ldots, T.$$

This corresponds to `Gmm("y",2,99)`, which runs from two lags, and then onwards as much as possible (99 will be larger than the number of time periods). For the unbalanced panel in the example:

$$Z_0 = \begin{pmatrix} y_{0,76} & y_{0,77} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & y_{0,76} & y_{0,77} & y_{0,78} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & y_{0,76} & y_{0,77} & y_{0,78} & y_{0,79} \end{pmatrix},$$

$$Z_1 = \begin{pmatrix} 0 & 0 & . & y_{0,77} & y_{0,78} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & . & y_{0,77} & y_{0,78} & y_{0,79} \end{pmatrix}.$$

The first columns (the dots) in Z_1 correspond to the missing equation, and are set to zero.

Several GMM-type instruments may be used, for example:

```
Gmm("y", 3, 99);
Gmm("y", 2, 2);
```

corresponds to the single call:

```
Gmm("y", 2, 99);
```

Similarly, it is possible to use future instruments, e.g.:

[10] Unlike the GAUSS version, the instruments are not actually created by `Gmm`. Instead, the instruments for each individual are created when required in the estimation procedure.

```
Gmm("y", 1, -1);
```

which corresponds to:

$$Z_0 = \begin{pmatrix} y_{0,76} & y_{0,77} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & y_{0,77} & y_{0,78} & y_{0,79} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & y_{0,78} & y_{0,79} & y_{0,80} \end{pmatrix},$$

$$Z_1 = \begin{pmatrix} 0 & 0 & 0 & y_{0,77} & y_{0,78} & y_{0,79} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & y_{0,78} & y_{0,79} & y_{0,80} \end{pmatrix}.$$

To use all observations use the following two calls:

```
Gmm("y", 2, 99);
Gmm("y", 1, -99);
```

DPD::GmmLevel

`GmmLevel(const sX, const iLag, const bDiff);`

`sX`	in:	string, name of variable for GMM-type instrument.
`iLag`	in:	m, where $t-m$ is most recent observation used as instrument.
`bDiff`	in:	`TRUE`: use differenced variable, else use level.

No return value.[11]

Description

Use this function to specify GMM-type instruments for the levels equation. This function is only relevant for GMM-SYS estimation where transformed and levels equations are combined. The differences or levels of the named variables are used depending on the last argument. The instruments are automatically adjusted for time periods which consist entirely of missing data. A call to `DeSelect` will also clear all instruments. Using `GmmLevel` switches to combined estimation automatically (also see `SetCombined`).
Using the example data given under the `Gmm` function, we could use the lagged difference of the dependent variable as instruments for the levels equation:

$$Z_i^+ = [\text{diag}(\Delta y_{i1}, \cdots, \Delta y_{iT-2})]\,.$$

This corresponds to `GmmLevel("y",1,1)`. In the example we have an extra observation for the first individual, and the actual instrument matrices are:

$$Z_0 = \begin{pmatrix} \Delta y_{0,77} & 0 & 0 \\ \Delta y_{0,78} & 0 & 0 \\ 0 & \Delta y_{0,79} & 0 \\ 0 & 0 & \Delta y_{0,80} \end{pmatrix}, \quad Z_1 = \begin{pmatrix} 0 & 0 & 0 \\ \Delta y_{0,78} & 0 & 0 \\ 0 & \Delta y_{0,79} & 0 \\ 0 & 0 & \Delta y_{0,80} \end{pmatrix}.$$

`GmmLevel("y",1,0)` is:

$$Z_0 = \begin{pmatrix} y_{0,76} & 0 & 0 \\ y_{0,77} & 0 & 0 \\ y_{0,78} & 0 & 0 \\ 0 & y_{0,79} & 0 \\ 0 & 0 & y_{0,80} \end{pmatrix}, \quad Z_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ y_{0,78} & 0 & 0 \\ 0 & y_{0,79} & 0 \\ 0 & 0 & y_{0,80} \end{pmatrix},$$

while `GmmLevel("y",2,1)` corresponds to:

$$Z_0 = \begin{pmatrix} \Delta y_{0,77} & 0 & 0 \\ 0 & \Delta y_{0,78} & 0 \\ 0 & 0 & \Delta y_{0,79} \end{pmatrix}, \quad Z_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \Delta y_{0,78} & 0 \\ 0 & 0 & \Delta y_{0,79} \end{pmatrix}.$$

[11] Unlike the GAUSS version, the instruments are not actually created by `GmmLevel`. Instead, the instruments for each individual are created when required in the estimation procedure.

DPD::InitData

`InitData()`

No return value.

Description

`InitData` is called by `Estimate` to initialize the data for estimation. The function calls `getData` to extract the data (`getData` creates `m_mGmm` to store the GMM-type variable), and `getDummies` to create the dummies. Transforms the data according to the specification given with `SetTransform`.

The relevant names in `m_asW` are adjusted for the transformations applied.

This function also sets `m_cObs` to the total number of observations used in the estimation. Without first-differenced or orthogonal transformations this equals the number of rows in `mn_mYears` and the data matrices: $\sum_{i=1}^{N} T_i$. With such transformations N observations are lost: $\sum_{i=1}^{N}(T_i - 1)$, except when combined transformed/levels equations estimation is used.

DPD::Lag

`Lag(const mX, const iLag);`

`mX`	in:	$O \times k$ matrix, where O is the database size.
`iLag`	in:	lag length.

Return value

Returns an $O \times k$ matrix with the lagged data. Observations which are lost are replaced by the missing value (NaN).

Description

The panel lag can only be taken if the index is known. The lag is computed in a similar fashion to the `Diff` function.

This function is not normally required, because a model is formulated by specifying lag lengths of variables.

DPD::LogLik

`LogLik(const vP, const adFunc, const avScore, const amHess);`

`vP`	in:	1×1 matrix, with current τ
`adFunc`	in:	address of variable
	out:	log-likelihood at τ
`avScore`	in:	should be 0
`amHess`	in:	should be 0

Return value

Returns 1 if the likelihood can be evaluated, 0 otherwise.

Description

Calls `SetGlsSigma2(1~1~`τ`)` and sets the transformation to `T_GLS` to evaluate the likelihood as in (3).

DPD::OrthDev, DPD::OrthDevToDiff

`OrthDev(const mX);`
`OrthDevToDiff(const mX);`

`mX`	in:	$O \times k$ matrix.

Return value

`OrthDev` returns a $O \times k$ matrix with `mX` transformed to orthogonal transformations.

`OrthDevToDiff` transforms an input matrix which is in orthogonal deviations into first differences.

Description

Orthogonal transformations takes each observation in deviation from the future means, together with a standardization. As a result, the last observation is lost. However, *the data are stored with a one period lag*, so that the missing observation is the first for each individual. This brings the transformation in line with first differences, allowing the same estimation code to be used.
Both require `m_mStartEnd`, so can only be called after `InitData`.

DPD::Output

```
Output();
```

No return value.

Description

Prints estimation results. Output is automatically printed after `Estimate`, unless switched off by `SetPrint(FALSE);` (this facility is used in Monte Carlo estimation).

DPD::Select

```
Select(const iGroup, const aSel);
```

`iGroup`	in:	int, group indicator: `Y_VAR`, `X_VAR`, `I_VAR` or `IL_VAR`
`aSel`	in:	array, specifying database name, start lag, end lag

No return value.

Description

Selects variables by name and with specified lags, and assigns the `iGroup` status to the selection. The `aSel` argument is an array consisting of sequences of three values: name, start lag, end lag. For examples, see §3.2.
The following types of variables are supported:

`Y_VAR`	dependent and lagged dependent variable
`X_VAR`	regressors which are not to be used as instruments
`I_VAR`	regressors which are used to construct the instrument matrix (first differencing or orthogonal deviations transformation is applied)
`IL_VAR`	regressors which are used to construct the instrument matrix (in levels: these are not transformated)

Each `Select` adds to the current selection. Use `DeSelect` to start afresh. *Note:* `SetSelSample` is not required, as the full sample will always be used (after dropping missing observations).
A model is formulated in levels, with differencing being left to DPD. If there are no instruments, the model is estimated by OLS.

DPD::SetCombined

```
SetCombined(const bCombined);
```

`bCombined`	in:	integer, `TRUE`: use GMM-SYS estimator, `FALSE`: normal estimation (is the default).

No return value.

Description

Used to switch to the Blundell and Bond (1998) estimator which combines transformed and levels equations. The `GmmLevel` function is used to formulate the GMM instruments for the levels equations. `GmmLevel` switches to combined estimation automatically, so normally there is no need to use `SetCombined`.

DPD::SetDummies

`SetDummies(const flDummy);`

`flDummy` in: integer, specifies the dummies to be included, see below.

No return value.

Description

Specifies the set of dummies to be used in the model. By default, a constant is used, which is equivalent to `SetDummies(D_CONSTANT)`. To use no dummies at all specify `SetDummies(D_NONE)`. Use `Select` prior to `SetDummies` to formulate the model.

The following types of dummies are supported:

`D_NONE`	(or 0) no dummies at all
`D_CONSTANT`	constant term
`D_TIME`	time dummies
`D_GROUP`	group dummies
`D_TIMEGROUP`	time dummies interacted with group dummies
`D_INDIVIDUAL`	individual dummies

Group dummies are only available if the group variable has been specified using `SetGroup`. Dummies can be combined by adding the values together, e.g. to use a constant, time and group dummies:

```
// ... code creating dpd and formulating model
dpd.SetDummies(D_CONSTANT+D_TIME+D_GROUP);
```

The constant is automatically deleted for within estimation.

DPD::SetGroup

`SetGroup(const sGroup);`

`sGroup` in: string, name of the column with a group index.

No return value.

Description

Specifies which column has the group variable. This function is only necessary when group dummies are required. The group variable should be an integer which specifies the group. The group indicator need not have a consecutive range (however, the more gaps, the longer it takes to construct the dummy), nor need the indicator start at 0 or 1.

DPD::SetGlsSigma2

`SetGlsSigma2(const vGlsSigma2)`

`vGlsSigma2` in: 1×3 matrix with σ_v^2, σ_a^2, and σ_η^2

No return value.

Description

Sets the variances to be used in the GLS transformation.

DPD::SetIndex

`SetIndex(const sIndex);`

`sIndex` in: string, name of the column with the individual index.

No return value.

Description

Specifies which column has the variable which identifies the individual (or company, etc.). This variable is used to determine the valid sample for estimation. The individual index should be a non-negative number. The `Diff` and `Lag` functions also require this information in order to lag data correctly, see §2.1.
Observations with a missing value (NaN) in the index are omitted. This allows dropping individuals altogether, for example:

```
indiv  =  indiv .== 1 .? M_NAN .: indiv;
```

will lead to the exclusion of the individual with index 1.
Using this function is recommended, but not compulsary. When omitted, a default index is constructed from the time information, also see under `SetYear`.

DPD::SetMethod

```
SetMethod(const iMethod);
```

`iMethod` in: int, estimation method.

No return value.

Description

Specifies the estimation method (also see §4.1 and §3):

`M_1STEP`	one-step estimation
`M_2STEP`	two-step estimation (also reports 1-step with robust std.errors)
`M_DPD_GLS`	Feasible GLS
`M_DPD_GLS1`	GLS (OLS residuals)
`M_DPD_ML`	Maximum likelihood (ML)

DPD::SetOptions

```
SetOptions(const fRobust, ...);
SetOptions(const fRobust, const bConcentrateD, const bTransformD,
    const bPrintGmm);
```

`fRobust` in: integer, `TRUE`: use robust variances in one-step estimator, `FALSE`: no robust variances (is the default) (-1 : argument is ignored).

`bConcentrateD` in: integer, `TRUE`: the dummies will be concentrated prior to estimation (-1 : argument is ignored)

`bTransformD` in: integer, `TRUE`: transform dummies, `FALSE`: dummies are used unchanged as instruments (is the default) (-1 : argument is ignored).

`bPrintGmm` in: integer, `TRUE`: print contents of GMM instruments for maximum T, `FALSE`: omit this information (is the default) (-1 : argument is ignored).

No return value.

Description

Specifies less-frequently used estimation options. `SetOptions` also calls `ClearModel`.
Robust one-step estimation is automatically reported as part of two-step estimation.
The second argument controls the estimation treatment of the selected set of dummies. This options can be used to have them concentrated out from the dependent variable, regressors and instruments prior to estimation. By default the dummies are not concentrated out. Concentration could reduce the memory requirements of the estimation problem, but is not quite exact when instrumental variable estimation is used.

DPD::SetPrint

```
SetPrint(fPrint);
```

`fPrint` in: int, `TRUE` or `FALSE`

No return value.

Description

Switches printing on (`TRUE`) or off (`FALSE`). By default printing is on, but for simulations it must be switched off.

DPD::SetSelSample

```
SetSelSample(const iYear1, const iPeriod1, const iYear2,
    const iPeriod2);
```

Description

This function is not used by DPD, unlike many other packages. By default, the whole sample is used, but DPD drops observations with missing values. This offers a convenient way to estimate over subsamples.

DPD::SetTest

```
SetTest(const iTestSpec, const iArOrder);
```

`iTestSpec` in: integer, 1: compute Wald and Sargan tests (the default); 2: only Sargan test; 0: neither.

`iArOrder` in: integer, compute AR tests up to specified order (0 by default).

No return value.

Description

Specifies test options.

DPD::SetTransform

```
SetTransform(const iTransform);
```

`iTransform` in: integer, specifies the transformation to be used in estimation.

No return value.

Description

The following types of transformations are supported:

`T_DIFFERENCES`	first differences
`T_DEVIATIONS`	orthogonal transformations
`T_WITHIN`	within group (deviations from individual means)
`T_NONE`	levels (no transformation)
`T_BETWEEN`	between group (using individual means)
`T_GLS`	GLS transformation (weighted deviations from individual means)

DPD::SetYear

```
SetYear(const sYear);
SetYear(const sYear, const sPeriod);
```

`sYear` in: string, name of the column with the years.

`sPeriod` in: string, optional argument for non-annual data: the name of the column with the periods.

No return value.

Description

Specifies which column has the year variable. This function is compulsary, and must be used after the data has been loaded, but before the model is formulated.

If seasonal data is used, the variable specified by the second argument should indicate the period for each observation. The earliest period should be 1; e.g. for quarterly data, the quarter should be either 1,2,3 or 4. If no individual variable is set yet (using `SetIndex`), this function will construct one from the year (and optional period) information. There are situations where this will not work correctly, see §2.1. Any subsequent calls to `SetIndex` will override the version constructed by `SetYear`. If the variable constructed by `SetYear` is used, this is noted in the output behind the number of individuals (as in §3.2).
Observations with a missing value (NaN) in the year are omitted. This allows dropping years altogether, for example:

```
newyear = year .< 1980 .? M_NAN .: year;
```

will lead to the exclusion of the observations before 1980.

DPD::SPrintTime

```
SPrintTime(const iT);
```

`iT` in: time index (0 for earliest in database).

Return value

Returns a string with the time index translated to a sample point.

DPD::TestWald

```
TestWald(const mWhich);
```

`mWhich` in: integer (0 or 1) or matrix.

Return value

Returns a 2×1 matrix with the test statistic in the first, and the p-value in the second row.

Description

Tests the significance of variables using the Wald test, depending on the `mWhich` argument:

`0`	tests all regressors except the dummies: 'Wald (joint)',
`1`	tests all dummies: 'Wald (dummy)',
`2`	tests the time dummies: 'Wald (time)',
$1 \times s$ matrix	specifies the indices of variables to test.

Remember that Ox indexing starts at 0, so `TestWald(<0:3>)` would test the significance of the first three variables in the model. Use `SetPrint` to switch printing of the test results on or off.

DPD::TimeAvg, DPD::TimeDev, DPD::TimeDevW

```
TimeAvg(const mX);
TimeDev(const mX);
TimeDevW(mX, const vGlsSigma2);
```

`mX` in: $O \times k$ matrix.

`vGlsSigma2` in: 1×3 matrix with σ_v^2, σ_a^2, and σ_η^2

Return value

`TimeAvg` returns an $N \times k$ matrix with the individual means (i.e. the means over time) of `mX`.
`TimeDev` returns an $O \times k$ matrix with `mX` in deviation from individual means.
`TimeDevW` returns an $O \times k$ matrix with `mX` in weighted deviation from individual means. The weights are determined by the values in `vGlsSigma2`, see (2).

Description

Requires `m_mStartEnd`, so can only be called after `InitData`.

C DPDSim: Monte Carlo experiments

The DPDSim class derives from Simulation. It implements a set of experiments as described in §3.4.

DPDSim::DPDSim

```
DPDSim::DPDSim(const cT, const cN, const cTdiscard,
    const dAlpha, const dBeta, const dRho, const dPhi,
    const dZsigma, const flDummies, const iTransform, iMethod,
    const bTest, const cRep);
```

cT	in:	int, T
cN	in:	int, N
cTdiscard	in:	int, number of observations to discard
dAlpha	in:	double, DGP value of α
dBeta	in:	double, DGP value of β
dRho	in:	double, DGP value of ρ
dPhi	in:	double, DGP value of ϕ
dZsigma	in:	double, DGP value of σ_e^2
flDummies	in:	int, dummy specification of model, see SetDummies
iTransform	in:	int, transformation for model estimation, see SetTransform
iMethod	in:	int, estimation method; in addition to the ones available in SetMethod (M_1STEP, M_2STEP, M_DPD_GLS, M_DPD_GLS1, M_DPD_ML), DPDSim also defines: M_AHd, M_AHl, M_1STEP_ROBUST, M_1STEP_COMBINED, M_2STEP_COMBINED
bTest	in:	int, TRUE: include t-tests, Sargan and AR tests
cRep	in:	int, number of replications M

No return value.

Description

Constructs a DPD simulation experiment. The notation refers to the DGP in §3.4. An example of its use is:

```
decl exp;
exp = new DPDSim(7, 100, 10, 0.5, 1, 0.8, 0.0, 0.9,
    D_CONSTANT, T_DIFFERENCES, M_1STEP, FALSE, 100);
exp.Simulate();
```

D DPD non-exported member functions

The functions documented here are only called from other DPD function members, indicated by a lower case first letter of the function name. The documentation is only included to help understand the code. Some functions are quite complex, and should be approached with care.

DPD::doEstimation

```
doEstimation(const iMethod, const bPrint)
```

No return value.

`iMethod`	in:	estimation method
`bPrint`	in:	if both `bPrint` and `m_fPrint` are `TRUE`: print output and do specification, Sargan and AR tests after estimation

Description

Estimation function called by `Estimate` and `EstimateGLS`.
When successful, `doEstimation` sets `m_iModelStatus` to `MS_ESTIMATED` and `m_iMethod` to the `iMethod` value used in the call.

DPD::doEstimationGls

```
doEstimationGls(const iMethod);
```

`iMethod`	in:	integer, one of: `M_DPD_GLS1`: GLS estimation based on OLS estimates `M_DPD_GLS`: GLS estimation based on within/between estimates `M_DPD_ML`: maximum likelihood estimation (iterated GLS)

No return value.

Description

Performs GLS estimation using prespecified variance ratios.
It is also possible to set the variances separately using `SetGlsSigma2` to define the weights in the GLS transformation. This can then be estimated using `SetTransform(T_GLS)` and `Estimate`.

DPD::doTests

```
doTests();
```

No return value.

Description

Runs the required test statistics (depending on `SetTest`.

DPD::getData

```
getData();
```

No return value.

Description

Extracts the data as specified (normally *in levels*) from the database prior to estimation. The status which was specified using `Select` is used here:

`m_mY`	`Y_VAR` variable with lag 0
`m_mW`	lagged `Y_VAR` variables and all `X_VAR` variables
`m_mI`	all `I_VAR` variables
`m_mIL`	all `IL_VAR` variables
`m_mD`	the dummy variables

First, these are extracted in raw form; next missing observations are dropped as follows:

(1) Construct a column vector `m_mYears`, with the years variable, and a row vector `mindex` with the index in the database of each observation.
(2) If a group column was specified, construct a column vector `m_mGroups`, with the groups.
(3) Construct a column vector `misobs`, with a zero for each observation without any missing value in any of the extracted data matrices.
(4) Drop all rows from `m_mYears` which have a one in the corresponding column in `misobs`; similarly drop columns of the remaining extracted data matrices.

Note that the model is formulated in levels. For estimation, a transformation such as differencing will be used. Differencing results in each first observation being lost. The starting and ending index of each individual is computed as discussed in §2.1.
Finally, sample size information is computed (all integers):

`m_mnTime`	earliest time index in sample,
`m_mxTime`	latest time index in sample,
`m_cT`	maximum number of time-series observations, $T = max\{T_i\}$,
`m_cN`	number of cross-section observations, N,

These are predifferencing sample sizes, which must be adjusted to get the estimation sample sizes: $O - N$ for the total number of observations, and $T - 1$ for the maximum time series (T for transformations other than differencing or orthogonal transformations; the adjustments are made in `InitData`).

DPD::getDummies

```
getDummies(const flDummy);
```

`flDummy` in: integer, specifies the dummies to be included, see below.

No return value.

Description

Creates the variable `m_mD` which holds the matrix with dummies (`m_cD` is the number of columns of `m_mD`). The `flDummy` argument is as passed to `Estimate`.
The information on which dummies are created is also stored in:

`m_fConstant`	`TRUE` if constant included
`m_cTimeDummy`	> 0: number of time dummies included
`m_cGroupDummy`	> 0: number of group dummies included
`m_cTimeGroupDummy`	> 0: number of time/group interaction dummies

The names of the dummies are appended to `m_asW`. The constant and group dummies are not created when the transformation is `T_WITHIN`.

DPD::getZSize, DPD::getZSizeLev

```
getZSize();
getZSizeLev(const bPrint);
```

Return value

The `getZSize` function returns the number of instrument columns z^* (see `getZti`).
The `getZSizeLev` function returns the number of instrument columns for the level equations z^+ (see `getZtiLev`).

DPD::getZti

```
DPD::getZti(const i, const mI, const cZ, const ai1, const ai2);
```

`i`	in:	int, index of individual.
`mI`	in:	$O \times m$ matrix, transformed $\boldsymbol{I}$, normal instruments.
`cZ`	in:	int, no of GMM-type instrument columns
`ai1`	in:	address of a variable.
	out:	starting observation.
`ai2`	in:	address of a variable.
	out:	ending observation.

No return value.

Description

The instrument matrix Z_i is created out the selections made using `Gmm` and the I_i (the variables specified using `I_VAR` and `IL_VAR` in `Select`). The `Gmm` information is stored in:

`m_mGmm`	in:	$O \times g$ matrix, variables for GMM-type instruments.
`m_mGmmInfo`	in:	$g \times 4$ matrix, with info for each variable: minimum lag, maximum lag length, first available observation, last available observation.

Constructing Z_i is a not straightforward, especially when the panel is unbalanced. Z_i consists mainly of zeros, and in each row there is a run of y values, starting in the column after the end of the previous run. When m (the longest lag on y) is 1, these runs are y_{i0}, $y_{i0}y_{i1}$, $y_{i0}y_{i1}y_{i2}$, etc. (indices start at zero here, as in Ox). When $m = 2$, the runs are $y_{i0}y_{i1}$, $y_{i0}y_{i1}y_{i2}$, $y_{i0}y_{i1}y_{i2}y_{i3}$, etc. In addition, there can be GMM instruments based on non-modelled variables; further examples are given under the `Gmm` function.

DPD::getZtiLev

```
DPD::getZtiLev(const i, const mI, const cZ, const ai1, const ai2);
```

`i`	in:	int, index of individual.
`mI`	in:	$O \times m$ matrix, transformed $\boldsymbol{I}$, normal instruments.
`cZ`	in:	int, no of GMM-type instrument columns for levels equations
`ai1`	in:	address of a variable.
	out:	starting observation.
`ai2`	in:	address of a variable.
	out:	ending observation.

No return value.

Description

The instrument matrix Z_i^+ for the levels equation (used in combined estimation) is created out the selections made using `GmmLevel` and dummies and the I_i (the variables specified using `I_VAR` in `Select`). The `Gmm` information is stored in:

`m_mGmmLev`	in:	$O \times g^+$ matrix, variables for GMM-type instruments.
`m_mGmmLevInfo`	in:	$g^+ \times 4$ matrix, with info for each variable: minimum lag, maximum lag length, first available observation, last available observation.

Examples are given under the `GmmLevel` function.

DPD::gmmProd1

```
gmmProd1(const fnGetZti, const cZ, const mY, const mW, const mI,
    const mH, const amYZ, const amWZ, const amZHZ);
```

`fnGetZti`	in:	function to construct $\boldsymbol{Z}_i$
`mY`	in:	$O \times 1$ matrix, transformed $\boldsymbol{Y}$.
`mW`	in:	$O \times k$ matrix, transformed $\boldsymbol{W}$.
`cZ`	in:	int, no of GMM-type instrument columns
`mI`	in:	$O \times m$ matrix, transformed $\boldsymbol{I}$, normal instruments.
`mH`	in:	$T \times T$ matrix, H, or 1 for identity matrix.
`amYZ`	in:	address of a variable.
	out:	$1 \times p$ matrix, $\sum_i Y_i' Z_i$.
`amWZ`	in:	address of a variable.
	out:	$k \times p$ matrix, $\sum_i W_i' Z_i$.
`amZHZ`	in:	address of a variable.
	out:	$p \times p$ matrix, $\sum_i Z_i' H Z_i$.

No return value.

Description

This function constructs some cross-product matrices involving the instruments. For each individual, Z_i is created using `fnGetZti`, and the requested products of data with Z_i are accumulated.

DPD::gmmProd2

```
gmmProd2(const fnGetZti, const cZ, const mI, const mV);
gmmProd2mix(const fnGetZti, const cZ, const mI, const mV,
    const fnGetZtilev, const cZlev, const mIlev, const mVlev);
```

`fnGetZti`	in:	function to construct $\boldsymbol{Z}_i$
`cZ`	in:	int, no of GMM-type instrument columns
`mI`	in:	$O \times m$ matrix, transformed $\boldsymbol{I}$, normal instruments.
`mV`	in:	0 or $O \times 1$ matrix, residuals $\boldsymbol{V}$.
`fnGetZtilev`	in:	function to construct $\boldsymbol{Z}_i^+$
`cZlev`	in:	int, no of GMM-type instrument columns in levels equations
`mIlev`	in:	$O \times m$ matrix, normal instruments.
`mVlev`	in:	0 or $O \times 1$ matrix, residuals $\boldsymbol{V}^+$.

Return value

`gmmProd2` returns $\sum_i Z_i' v_i v_i' Z_i$. `gmmProd2mix` returns $\sum_i Z_i' v_i v_i' Z_i$, where Z_i consists of both Z_i^* and Z_i^+.

Description

This function constructs the second-step optimal weight matrix $\boldsymbol{A}_N$ involving the instruments and residuals.

DPD::gmmProd3

```
gmmProd3(const fnGetZti, const cZ, const mI, const mH, const mW,
    const amZHw);
```

`fnGetZti`	in:	function to construct $\boldsymbol{Z}_i$
`cZ`	in:	int, no of GMM-type instrument columns
`mI`	in:	$O \times m$ matrix, transformed $\boldsymbol{I}$, normal instruments.
`mH`	in:	$T \times T$ matrix, H, or 1 for identity matrix.
`mW`	in:	0 or $O \times 1$ matrix, lagged residuals $\boldsymbol{W} = \boldsymbol{V}_{-m}$.
`amZHw`	in:	0 or address of a variable.
	out:	$1 \times p$ matrix, $\sum_i Z_i' H w_i$.

Return value

Returns $\sum_i (w_i' H w_i)^2$.

Description

This function constructs the cross-product matrices involving the instruments and residuals and lagged residuals as needed in the AR test after one-step estimation.

DPD::gmmProd4

```
gmmProd4(const fnGetZti, const cZ, const mI, const mV, const mU,
    const mW, const amZVs);
gmmProd4mix(const fnGetZti, const cZ, const mI, const mV, const mU,
    const mW, const fnGetZtilev, const cZlev, const mIlev,
    const mVlev, const amZVs)
```

fnGetZti	in:	function to construct $\boldsymbol{Z}_i$
cZ	in:	int, no of GMM-type instrument columns
mI	in:	$O \times m$ matrix, transformed $\boldsymbol{I}$, normal instruments.
mV	in:	0 or $O \times 1$ matrix, residuals $\boldsymbol{V}$.
mU	in:	0 or $O \times 1$ matrix, $\boldsymbol{U}$.
mW	in:	0 or $O \times 1$ matrix, lagged residuals $\boldsymbol{W} = \boldsymbol{V}_{-m}$.
amZVs	in:	0 or address of a variable.
	out:	$1 \times p$ matrix, gmmProd4: $\sum_i Z_i' v_i (u_i' w_i)$. gmmProd4mix, the same but with Z_i consisting of both Z_i^* and Z_i^+.

Return value

Returns $\sum_i (u_i' w_i)^2$.

Description

This function constructs the cross-product matrices involving the instruments and residuals and lagged residuals as needed in the AR test after two-step estimation.

DPD::testAr

```
testAr(/* many arguments */);
```

Return value

Returns a 2×1 matrix with the test statistic in the first, and the p-value in the second row.

Description

Tests the model for serial correlation using the test described in equations (8) and (9) of Arellano and Bond (1991, p.282). Use SetPrint to switch printing of the test results on or off.

This function is called after estimation when requested. Use SetTest to specify the calculation of AR tests.

DPD::testSargan

```
testSargan(const mYZ, const mWZ, const mAN);
```

mYZ	in:	$k \times p$ matrix, $\boldsymbol{Y}'\boldsymbol{Z} = \sum_i Y_i' Z_i$.
mWZ	in:	$k \times p$ matrix, see testAr.
mAN	in:	$p \times p$ matrix, A, see testAr

Return value

Returns a 2×1 matrix with the test statistic in the first, and the p-value in the second row.

Description

Performs Sargan's test of overidentifying restrictions. For two-step estimation, the test is based on equation (10) of Arellano and Bond (1991, p.282). After one-step estimation, the test is based on s_1, which is just given below (10) in Arellano and Bond (1991). Sargan's test is not reported after one-step robust estimation. Use SetPrint to switch printing of the test results on or off.

This function is called after estimation when requested. Use SetTest to specify the calculation of this test.