1 Purpose

nag_pls_orth_scores_wold (g02lbc) fits an orthogonal scores partial least squares (PLS) regression by using Wold’s iterative method.

2 Specification

```c
#include <nag.h>
#include <nagg02.h>

void nag_pls_orth_scores_wold (Nag_OrderType order, Integer n, Integer mx, const double x[], Integer pdx, const Integer isx[], Integer ip, Integer my, const double y[], Integer pdy, double xbar[], double ybar[], Nag_ScalePredictor iscale, double xstd[], double ystd[], Integer maxfac, Integer maxit, double tau, double xres[], Integer pdxres, double yres[], double w[], Integer pdw, double p[], Integer pdp, double t[], Integer pdt, double c[], Integer pdc, double u[], Integer pdu, double xcv[], Integer pdxcv, NagError *fail)
```

3 Description

Let $X_1$ be the mean-centred $n$ by $m$ data matrix $X$ of $n$ observations on $m$ predictor variables. Let $Y_1$ be the mean-centred $n$ by $r$ data matrix $Y$ of $n$ observations on $r$ response variables.

The first of the $k$ factors PLS methods extract from the data predicts both $X_1$ and $Y_1$ by regressing on a $t_1$ column vector of $n$ scores:

$$
\hat{X}_1 = t_1 p_1^T, \quad \hat{Y}_1 = t_1 c_1^T, \quad \text{with} \quad t_1^T t_1 = 1,
$$

where the column vectors of $m$ $x$-loadings $p_1$ and $r$ $y$-loadings $c_1$ are calculated in the least squares sense:

$$
p_1^T = t_1^T X_1, \quad c_1^T = t_1^T Y_1.
$$

The $x$-score vector $t_1 = X_1 w_1$ is the linear combination of predictor data $X_1$ that has maximum covariance with the $y$-scores $u_1 = Y_1 c_1$, where the $x$-weights vector $w_1$ is the normalised first left singular vector of $X_1^T Y_1$.

The method extracts subsequent PLS factors by repeating the above process with the residual matrices:

$$
X_i = X_{i-1} - \hat{X}_{i-1}, \quad Y_i = Y_{i-1} - \hat{Y}_{i-1}, \quad i = 2, 3, \ldots, k,
$$

and with orthogonal scores:

$$
t_i^T t_j = 0, \quad j = 1, 2, \ldots, i - 1.
$$

Optionally, in addition to being mean-centred, the data matrices $X_1$ and $Y_1$ may be scaled by standard deviations of the variables. If data are supplied mean-centred, the calculations are not affected within numerical accuracy.

4 References

5 Arguments

1: \texttt{order} – Nag_OrderType \hfill \textit{Input}

\textit{On entry}: the \texttt{order} argument specifies the two-dimensional storage scheme being used, i.e., row-major ordering or column-major ordering. C language defined storage is specified by \texttt{order} = Nag_RowMajor. See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

\textit{Constraint}: \texttt{order} = Nag_RowMajor or Nag_ColMajor.

2: \texttt{n} – Integer \hfill \textit{Input}

\textit{On entry}: \texttt{n}, the number of observations.

\textit{Constraint}: \texttt{n} > 1.

3: \texttt{mx} – Integer \hfill \textit{Input}

\textit{On entry}: the number of predictor variables.

\textit{Constraint}: \texttt{mx} > 1.

4: \texttt{x[dim]} – const double \hfill \textit{Input}

\textit{Note}: the dimension, \textit{dim}, of the array \texttt{x} must be at least

\begin{align*}
\max(1, \texttt{pdx} \times \texttt{mx}) & \text{ when } \texttt{order} = \text{Nag_ColMajor}; \\
\max(1, \texttt{n} \times \texttt{pdx}) & \text{ when } \texttt{order} = \text{Nag_RowMajor}.
\end{align*}

Where \( \text{X}(i,j) \) appears in this document, it refers to the array element

\begin{align*}
\texttt{x}[\textit{dim}] \times \texttt{pdx} + i - 1 & \text{ when } \texttt{order} = \text{Nag_ColMajor}; \\
\texttt{x}[\textit{dim}] \times \texttt{pdx} + j - 1 & \text{ when } \texttt{order} = \text{Nag_RowMajor}.
\end{align*}

\textit{On entry}: \text{X}(i,j) must contain the \textit{i}th observation on the \textit{j}th predictor variable, for \textit{i} = 1, 2, \ldots, \texttt{n} and \textit{j} = 1, 2, \ldots, \texttt{mx}.

5: \texttt{pdx} – Integer \hfill \textit{Input}

\textit{On entry}: the stride separating row or column elements (depending on the value of \texttt{order}) in the array \texttt{x}.

\textit{Constraints}:

\begin{align*}
\text{if } \texttt{order} = \text{Nag_ColMajor}, & \texttt{pdx} \geq \texttt{n}; \\
\text{if } \texttt{order} = \text{Nag_RowMajor}, & \texttt{pdx} \geq \texttt{mx}.
\end{align*}

6: \texttt{isx[mx]} – const Integer \hfill \textit{Input}

\textit{On entry}: indicates which predictor variables are to be included in the model.

\texttt{isx}[\textit{j} - 1] = 1

The \textit{j}th predictor variable (with variates in the \textit{j}th column of \text{X}) is included in the model.

\texttt{isx}[\textit{j} - 1] = 0

Otherwise.

\textit{Constraint}: the sum of elements in \texttt{isx} must equal \texttt{ip}.

7: \texttt{ip} – Integer \hfill \textit{Input}

\textit{On entry}: \texttt{m}, the number of predictor variables in the model.

\textit{Constraint}: \texttt{1} < \texttt{ip} \leq \texttt{mx}.
8: \textit{my} – Integer \hspace{1cm} \textit{Input}

\textit{On entry}: \( r \), the number of response variables.

\textit{Constraint}: \( my \geq 1 \).

9: \( y[dim] \) – const double \hspace{1cm} \textit{Input}

\textit{Note}: the dimension, \( dim \), of the array \( y \) must be at least
\[ \max(1, pdy \times my) \] when \( \text{order} = \text{Nag\_ColMajor}; \]
\[ \max(1, n \times pdy) \] when \( \text{order} = \text{Nag\_RowMajor}. \)

Where \( Y(i,j) \) appears in this document, it refers to the array element
\[ y[(j - 1) \times pdy + i - 1] \] when \( \text{order} = \text{Nag\_ColMajor}; \]
\[ y[(i - 1) \times pdy + j - 1] \] when \( \text{order} = \text{Nag\_RowMajor}. \)

\textit{On entry}: \( Y(i,j) \) must contain the \( i \)th observation for the \( j \)th response variable, for \( i = 1,2,\ldots,n \) and \( j = 1,2,\ldots,my \).

10: \( pdy \) – Integer \hspace{1cm} \textit{Input}

\textit{On entry}: the stride separating row or column elements (depending on the value of \( \text{order} \)) in the array \( y \).

\textit{Constraints}:
\[
\begin{align*}
\text{if } \text{order} = \text{Nag\_ColMajor}, & \quad pdy \geq n; \\
\text{if } \text{order} = \text{Nag\_RowMajor}, & \quad pdy \geq my.
\end{align*}
\]

11: \( xbar[ip] \) – double \hspace{1cm} \textit{Output}

\textit{On exit}: mean values of predictor variables in the model.

12: \( ybar[my] \) – double \hspace{1cm} \textit{Output}

\textit{On exit}: the mean value of each response variable.

13: \( \text{iscale} \) – \text{Nag\_ScalePredictor} \hspace{1cm} \textit{Input}

\textit{On entry}: indicates how predictor variables are scaled.

\( \text{iscale} = \text{Nag\_PredStdScale} \)
Data are scaled by the standard deviation of variables.

\( \text{iscale} = \text{Nag\_PredUserScale} \)
Data are scaled by user-supplied scalings.

\( \text{iscale} = \text{Nag\_PredNoScale} \)
No scaling.

\textit{Constraint}: \( \text{iscale} = \text{Nag\_PredNoScale}, \text{Nag\_PredStdScale} \) or \( \text{Nag\_PredUserScale} \).

14: \( xstd[ip] \) – double \hspace{1cm} \textit{Input/Output}

\textit{On entry}: if \( \text{iscale} = \text{Nag\_PredUserScale} \), \( xstd[j - 1] \) must contain the user-supplied scaling for the \( j \)th predictor variable in the model, for \( j = 1,2,\ldots,ip \). Otherwise \( xstd \) need not be set.

\textit{On exit}: if \( \text{iscale} = \text{Nag\_PredStdScale} \), standard deviations of predictor variables in the model. Otherwise \( xstd \) is not changed.

15: \( ystd[my] \) – double \hspace{1cm} \textit{Input/Output}

\textit{On entry}: if \( \text{iscale} = \text{Nag\_PredUserScale} \), \( ystd[j - 1] \) must contain the user-supplied scaling for the \( j \)th response variable in the model, for \( j = 1,2,\ldots,my \). Otherwise \( ystd \) need not be set.
On exit: if iscale = Nag_PredStdScale, the standard deviation of each response variable. Otherwise ystd is not changed.

16: **maxfac** – Integer

*Input*

On entry: \( k \), the number of latent variables to calculate.

*Constraint*: \( 1 \leq \text{maxfac} \leq \text{ip} \).

17: **maxit** – Integer

*Input*

On entry: if my = 1, maxit is not referenced; otherwise the maximum number of iterations used to calculate the \( x \)-weights.

*Suggested value*: maxit = 200.

*Constraint*: if my > 1, maxit > 1.

18: **tau** – double

*Input*

On entry: if my = 1, tau is not referenced; otherwise the iterative procedure used to calculate the \( x \)-weights will halt if the Euclidean distance between two subsequent estimates is less than or equal to tau.

*Suggested value*: tau = 1.0e−4.

*Constraint*: if my > 1, tau > 0.0.

19: **xres[\text{dim}]** – double

*Output*

*Note*: the dimension, \( \text{dim} \), of the array xres must be at least

\[
\max(1, \text{pdxres} \times \text{ip}) \quad \text{when order = Nag_ColMajor};
\]
\[
\max(1, n \times \text{pdxres}) \quad \text{when order = Nag_RowMajor}.
\]

The \((i, j)\)th element of the matrix is stored in

\[
xres[j - 1 \times \text{pdxres} + i - 1] \quad \text{when order = Nag_ColMajor};
\]
\[
xres[i - 1 \times \text{pdxres} + j - 1] \quad \text{when order = Nag_RowMajor}.
\]

On exit: the predictor variables’ residual matrix \( X_k \).

20: **pdxres** – Integer

*Input*

On entry: the stride separating row or column elements (depending on the value of order) in the array xres.

*Constraints*:

if order = Nag_ColMajor, pdxres \( \geq \text{n}; \)

if order = Nag_RowMajor, pdxres \( \geq \text{ip} \).

21: **yres[\text{dim}]** – double

*Output*

*Note*: the dimension, \( \text{dim} \), of the array yres must be at least

\[
\max(1, \text{pdyres} \times \text{my}) \quad \text{when order = Nag_ColMajor};
\]
\[
\max(1, n \times \text{pdyres}) \quad \text{when order = Nag_RowMajor}.
\]

The \((i, j)\)th element of the matrix is stored in

\[
yres[j - 1 \times \text{pdyres} + i - 1] \quad \text{when order = Nag_ColMajor};
\]
\[
yres[i - 1 \times \text{pdyres} + j - 1] \quad \text{when order = Nag_RowMajor}.
\]

On exit: the residuals for each response variable, \( Y_k \).
22: **pdyres** – Integer

*Input*

*On entry:* the stride separating row or column elements (depending on the value of **order**) in the array **yres**.

*Constraints:*

- if **order** = Nag_ColMajor, **pdyres** ≥ **n**;
- if **order** = Nag_RowMajor, **pdyres** ≥ **my**.

23: **w[**dim]**] – double

*Output*

*Note:* the dimension, **dim**, of the array **w** must be at least

\[
\max(1, \text{pdw} \times \text{maxfac}) \quad \text{when} \quad \text{order} = \text{Nag\_ColMajor};
\]

\[
\max(1, \text{ip} \times \text{pdw}) \quad \text{when} \quad \text{order} = \text{Nag\_RowMajor}.
\]

The \((i, j)\)th element of the matrix **W** is stored in

\[
\begin{align*}
\text{w}[(j - 1) \times \text{pdw} + i - 1] \quad \text{when} \quad \text{order} = \text{Nag\_ColMajor}; \\
\text{w}[(i - 1) \times \text{pdw} + j - 1] \quad \text{when} \quad \text{order} = \text{Nag\_RowMajor}.
\end{align*}
\]

*On exit:* the \(j\)th column of **W** contains the \(x\)-weights **w**\(_j\), for \(j = 1, 2, \ldots, \text{maxfac}\).

24: **pdw** – Integer

*Input*

*On entry:* the stride separating row or column elements (depending on the value of **order**) in the array **w**.

*Constraints:*

- if **order** = Nag_ColMajor, **pdw** ≥ **ip**;
- if **order** = Nag_RowMajor, **pdw** ≥ **maxfac**.

25: **p[**dim]**] – double

*Output*

*Note:* the dimension, **dim**, of the array **p** must be at least

\[
\max(1, \text{pdp} \times \text{maxfac}) \quad \text{when} \quad \text{order} = \text{Nag\_ColMajor};
\]

\[
\max(1, \text{ip} \times \text{pdp}) \quad \text{when} \quad \text{order} = \text{Nag\_RowMajor}.
\]

The \((i, j)\)th element of the matrix **P** is stored in

\[
\begin{align*}
\text{p}[(j - 1) \times \text{pdp} + i - 1] \quad \text{when} \quad \text{order} = \text{Nag\_ColMajor}; \\
\text{p}[(i - 1) \times \text{pdp} + j - 1] \quad \text{when} \quad \text{order} = \text{Nag\_RowMajor}.
\end{align*}
\]

*On exit:* the \(j\)th column of **P** contains the \(x\)-loadings **p**\(_j\), for \(j = 1, 2, \ldots, \text{maxfac}\).

26: **pdp** – Integer

*Input*

*On entry:* the stride separating row or column elements (depending on the value of **order**) in the array **p**.

*Constraints:*

- if **order** = Nag_ColMajor, **pdp** ≥ **ip**;
- if **order** = Nag_RowMajor, **pdp** ≥ **maxfac**.

27: **t[**dim]**] – double

*Output*

*Note:* the dimension, **dim**, of the array **t** must be at least

\[
\max(1, \text{pdt} \times \text{maxfac}) \quad \text{when} \quad \text{order} = \text{Nag\_ColMajor};
\]

\[
\max(1, \text{n} \times \text{pdt}) \quad \text{when} \quad \text{order} = \text{Nag\_RowMajor}.
\]

The \((i, j)\)th element of the matrix **T** is stored in

\[
\begin{align*}
\text{t}[(j - 1) \times \text{pdt} + i - 1] \quad \text{when} \quad \text{order} = \text{Nag\_ColMajor}; \\
\text{t}[(i - 1) \times \text{pdt} + j - 1] \quad \text{when} \quad \text{order} = \text{Nag\_RowMajor}.
\end{align*}
\]
On exit: the $j$th column of $T$ contains the $x$-scores $t_j$, for $j = 1, 2, \ldots, \text{maxfac}$.

28: \textbf{pdt} – Integer \hspace{1cm} \textit{Input}

On entry: the stride separating row or column elements (depending on the value of \textbf{order}) in the array \textbf{t}.

\textbf{Constraints}:
- if \textbf{order} = Nag_ColMajor, \textbf{pdt} $\geq n$;
- if \textbf{order} = Nag_RowMajor, \textbf{pdt} $\geq \text{maxfac}$.

29: \textbf{c[\text{dim}]} – double \hspace{1cm} \textit{Output}

\textbf{Note}: the dimension, \text{dim}, of the array \textbf{c} must be at least
- $\max(1, \text{pdc} \times \text{maxfac})$ when \textbf{order} = Nag_ColMajor;
- $\max(1, \text{my} \times \text{pdc})$ when \textbf{order} = Nag_RowMajor.

The $(i, j)$th element of the matrix $C$ is stored in
- $c[(j - 1) \times \text{pdc} + i - 1]$ when \textbf{order} = Nag_ColMajor;
- $c[(i - 1) \times \text{pdc} + j - 1]$ when \textbf{order} = Nag_RowMajor.

On exit: the $j$th column of $C$ contains the $y$-loadings $c_j$, for $j = 1, 2, \ldots, \text{maxfac}$.

30: \textbf{pdc} – Integer \hspace{1cm} \textit{Input}

On entry: the stride separating row or column elements (depending on the value of \textbf{order}) in the array \textbf{c}.

\textbf{Constraints}:
- if \textbf{order} = Nag_ColMajor, \textbf{pdc} $\geq \text{my}$;
- if \textbf{order} = Nag_RowMajor, \textbf{pdc} $\geq \text{maxfac}$.

31: \textbf{u[\text{dim}]} – double \hspace{1cm} \textit{Output}

\textbf{Note}: the dimension, \text{dim}, of the array \textbf{u} must be at least
- $\max(1, \text{pdu} \times \text{maxfac})$ when \textbf{order} = Nag_ColMajor;
- $\max(1, \text{n} \times \text{pdu})$ when \textbf{order} = Nag_RowMajor.

The $(i, j)$th element of the matrix $U$ is stored in
- $u[(j - 1) \times \text{pdu} + i - 1]$ when \textbf{order} = Nag_ColMajor;
- $u[(i - 1) \times \text{pdu} + j - 1]$ when \textbf{order} = Nag_RowMajor.

On exit: the $j$th column of $U$ contains the $y$-scores $u_j$, for $j = 1, 2, \ldots, \text{maxfac}$.

32: \textbf{pdu} – Integer \hspace{1cm} \textit{Input}

On entry: the stride separating row or column elements (depending on the value of \textbf{order}) in the array \textbf{u}.

\textbf{Constraints}:
- if \textbf{order} = Nag_ColMajor, \textbf{pdu} $\geq \text{n}$;
- if \textbf{order} = Nag_RowMajor, \textbf{pdu} $\geq \text{maxfac}$.

33: \textbf{xcv[\text{maxfac}]} – double \hspace{1cm} \textit{Output}

On exit: $\text{xcv}[j - 1]$ contains the cumulative percentage of variance in the predictor variables explained by the first $j$ factors, for $j = 1, 2, \ldots, \text{maxfac}$.
34: \( \text{ycv}[\text{dim}] \) – double

**Note:** the dimension, \( \text{dim} \), of the array \( \text{ycv} \) must be at least

\[
\max(1, \text{pdycv} \times \text{my}) \quad \text{when order = Nag\_ColMajor;}
\]

\[
\max(1, \text{maxfac} \times \text{pdycv}) \quad \text{when order = Nag\_RowMajor.}
\]

Where \( YCV(i,j) \) appears in this document, it refers to the array element

\[
\text{ycv}[(j-1) \times \text{pdycv} + i - 1] \quad \text{when order = Nag\_ColMajor;}
\]

\[
\text{ycv}[(i-1) \times \text{pdycv} + j - 1] \quad \text{when order = Nag\_RowMajor.}
\]

**On exit:** \( YCV(i,j) \) is the cumulative percentage of variance of the \( j \)th response variable explained by the first \( i \) factors, for \( i = 1, 2, \ldots, \text{maxfac} \) and \( j = 1, 2, \ldots, \text{my} \).

35: \( \text{pdycv} \) – Integer

**Input**

**On entry:** the stride separating row or column elements (depending on the value of \( \text{order} \)) in the array \( \text{ycv} \).

**Constraints:**

- if \( \text{order} = \text{Nag\_ColMajor}, \text{pdycv} \geq \text{maxfac}; \)
- if \( \text{order} = \text{Nag\_RowMajor}, \text{pdycv} \geq \text{my}. \)

36: \( \text{fail} \) – NagError*

**Input/Output**

The NAG error argument (see Section 3.6 in the Essential Introduction).

### 6 Error Indicators and Warnings

**NE_ALLOC_FAIL**

Dynamic memory allocation failed.

See Section 3.2.1.2 in the Essential Introduction for further information.

**NE_BAD_PARAM**

On entry, argument \( \langle\text{value}\rangle \) had an illegal value.

**NE_INT**

**On entry,** \( \text{mx} = \langle\text{value}\rangle. \)

Constraint: \( \text{mx} > 1. \)

**On entry,** \( \text{my} = \langle\text{value}\rangle. \)

Constraint: \( \text{my} \geq 1. \)

**On entry,** \( \text{n} = \langle\text{value}\rangle. \)

Constraint: \( \text{n} > 1. \)

**On entry,** \( \text{pdc} = \langle\text{value}\rangle. \)

Constraint: \( \text{pdc} > 0. \)

**On entry,** \( \text{pdp} = \langle\text{value}\rangle. \)

Constraint: \( \text{pdp} > 0. \)

**On entry,** \( \text{pdt} = \langle\text{value}\rangle. \)

Constraint: \( \text{pdt} > 0. \)

**On entry,** \( \text{pdu} = \langle\text{value}\rangle. \)

Constraint: \( \text{pdu} > 0. \)

**On entry,** \( \text{pdw} = \langle\text{value}\rangle. \)

Constraint: \( \text{pdw} > 0. \)

**On entry,** \( \text{pdx} = \langle\text{value}\rangle. \)

Constraint: \( \text{pdx} > 0. \)
On entry, \(pd\text{x}r\text{es} = \langle\text{value}\rangle\).
Constraint: \(pd\text{x}r\text{es} > 0\).

On entry, \(pd\text{y} = \langle\text{value}\rangle\).
Constraint: \(pd\text{y} > 0\).

On entry, \(pd\text{ycv} = \langle\text{value}\rangle\).
Constraint: \(pd\text{ycv} > 0\).

On entry, \(pd\text{y}r\text{es} = \langle\text{value}\rangle\).
Constraint: \(pd\text{y}r\text{es} > 0\).

\[\text{NE_INT}_2\]

On entry, \(ip = \langle\text{value}\rangle\) and \(mx = \langle\text{value}\rangle\).
Constraint: \(1 < ip \leq mx\).

On entry, \(ma\text{xfac} = \langle\text{value}\rangle\) and \(ip = \langle\text{value}\rangle\).
Constraint: \(1 \leq ma\text{xfac} \leq ip\).

On entry, \(my = \langle\text{value}\rangle\) and \(ma\text{xit} = \langle\text{value}\rangle\).
Constraint: if \(my > 1\), \(ma\text{xit} > 1\).

On entry, \(pd\text{c} = \langle\text{value}\rangle\) and \(ma\text{xfac} = \langle\text{value}\rangle\).
Constraint: \(pd\text{c} \geq ma\text{xfac}\).

On entry, \(pd\text{c} = \langle\text{value}\rangle\) and \(my = \langle\text{value}\rangle\).
Constraint: \(pd\text{c} \geq my\).

On entry, \(pd\text{p} = \langle\text{value}\rangle\) and \(ip = \langle\text{value}\rangle\).
Constraint: \(pd\text{p} \geq ip\).

On entry, \(pd\text{p} = \langle\text{value}\rangle\) and \(ma\text{xfac} = \langle\text{value}\rangle\).
Constraint: \(pd\text{p} \geq ma\text{xfac}\).

On entry, \(pd\text{t} = \langle\text{value}\rangle\) and \(ma\text{xfac} = \langle\text{value}\rangle\).
Constraint: \(pd\text{t} \geq ma\text{xfac}\).

On entry, \(pd\text{t} = \langle\text{value}\rangle\) and \(n = \langle\text{value}\rangle\).
Constraint: \(pd\text{t} \geq n\).

On entry, \(pd\text{u} = \langle\text{value}\rangle\) and \(ma\text{xfac} = \langle\text{value}\rangle\).
Constraint: \(pd\text{u} \geq ma\text{xfac}\).

On entry, \(pd\text{u} = \langle\text{value}\rangle\) and \(n = \langle\text{value}\rangle\).
Constraint: \(pd\text{u} \geq n\).

On entry, \(pd\text{w} = \langle\text{value}\rangle\) and \(ip = \langle\text{value}\rangle\).
Constraint: \(pd\text{w} \geq ip\).

On entry, \(pd\text{w} = \langle\text{value}\rangle\) and \(ma\text{xfac} = \langle\text{value}\rangle\).
Constraint: \(pd\text{w} \geq ma\text{xfac}\).

On entry, \(pd\text{x} = \langle\text{value}\rangle\) and \(mx = \langle\text{value}\rangle\).
Constraint: \(pd\text{x} \geq mx\).

On entry, \(pd\text{x} = \langle\text{value}\rangle\) and \(n = \langle\text{value}\rangle\).
Constraint: \(pd\text{x} \geq n\).

On entry, \(pd\text{x}r\text{es} = \langle\text{value}\rangle\) and \(ip = \langle\text{value}\rangle\).
Constraint: \(pd\text{x}r\text{es} \geq ip\).

On entry, \(pd\text{x}r\text{es} = \langle\text{value}\rangle\) and \(n = \langle\text{value}\rangle\).
Constraint: \(pd\text{x}r\text{es} \geq n\).

On entry, \(pd\text{y} = \langle\text{value}\rangle\) and \(my = \langle\text{value}\rangle\).
Constraint: \(pd\text{y} \geq my\).

On entry, \(pd\text{y} = \langle\text{value}\rangle\) and \(n = \langle\text{value}\rangle\).
Constraint: \(pd\text{y} \geq n\).
On entry, $pdycv = \langle value \rangle$ and $maxfac = \langle value \rangle$.
Constraint: $pdycv \geq maxfac$.

On entry, $pdycv = \langle value \rangle$ and $my = \langle value \rangle$.
Constraint: $pdycv \geq my$.

On entry, $pdyn = \langle value \rangle$ and $my = \langle value \rangle$.
Constraint: $pdyn \geq my$.

On entry, $pdyn = \langle value \rangle$ and $n = \langle value \rangle$.
Constraint: $pdyn < n$.

**NE_INT_ARG_CONS**
On entry, $ip$ is not equal to the sum of $isx$ elements: $ip = \langle value \rangle$, sum($isx$) = $\langle value \rangle$.

**NE_INT_ARRAY_VAL_1_OR_2**
On entry, element $\langle value \rangle$ of $isx$ is invalid.

**NE_INTERNAL_ERROR**
An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.
An unexpected error has been triggered by this function. Please contact NAG.
See Section 3.6.6 in the Essential Introduction for further information.

**NE_NO_LICENCE**
Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in the Essential Introduction for further information.

**NE_REAL**
On entry, $tau = \langle value \rangle$.
Constraint: if $my > 1$, $tau > 0.0$.

7 **Accuracy**
In general, the iterative method used in the calculations is less accurate (but faster) than the singular value decomposition approach adopted by nag_pls_orth_scores_svd (g02lac).

8 **Parallelism and Performance**
nag_pls_orth_scores_wold (g02lbc) is not threaded by NAG in any implementation.
nag_pls_orth_scores_wold (g02lbc) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.
Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users’ Note for your implementation for any additional implementation-specific information.

9 **Further Comments**
nag_pls_orth_scores_wold (g02lbc) allocates internally $(n + r)$ elements of double storage.

10 **Example**
This example reads in data from an experiment to measure the biological activity in a chemical compound, and a PLS model is estimated.
10.1 Program Text

/* nag_pls_orth_scores_wold (g02lbc) Example Program.
 * Copyright 2014 Numerical Algorithms Group.
 * Mark 9, 2009.
 */

#include <stdio.h>
#include <math.h>
#include <nag.h>
#include <nagg02.h>
#include <nagx04.h>

int main(void)
{
    /* Integer scalar and array declarations */
    Integer exit_status = 0;
    Integer i, ip, j, maxfac, maxit, mx, my, n;
    Integer pdc, pdp, pdt, pdu, pdw, pdx, pdxres, pdy, pdycv, pdyres;
    Integer *isx = 0;
    /* Double scalar and array declarations */
    double tau;
    double *c = 0, *p = 0, *t = 0, *u = 0, *w = 0, *x = 0, *xbar = 0;
    double *xcv = 0, *xres = 0, *xstd = 0, *y = 0, *ybar = 0;
    double *ycv = 0, *yres = 0, *ystd = 0;
    /* Character scalar and array declarations */
    char siscale[40];
    /* NAG Types */
    Nag_OrderType order;
    Nag_ScalePredictor iscale;
    NagError fail;

    INIT_FAIL(fail);
    printf("nag_pls_orth_scores_wold (g02lbc) Example Program Results\n");

    /* Skip header in data file. */
    #ifdef _WIN32
    scanf_s("%*[\n] ");
    #else
    scanf("%*[\n] ");
    #endif

    /* Read data values. */
    #ifdef _WIN32
    scanf_s("%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%39s %"NAG_IFMT"%[\n] ",
            &n, &mx, &my, siscale, _countof(siscale), &maxfac);
    #else
    scanf("%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%39s %"NAG_IFMT"%[\n] ",
            &n, &mx, &my, siscale, &maxfac);
    #endif
    iscale = (Nag_ScalePredictor) nag_enum_name_to_value(siscale);

    if (!(isx = NAG_ALLOC(mx, Integer)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
    for (j = 0; j < mx; j++)
    #ifdef _WIN32
    scanf_s("%"NAG_IFMT" ", &isx[j]);
    #else
    scanf("%"NAG_IFMT" ", &isx[j]);
    #endif
    #ifdef _WIN32
    scanf("%*[\n] ");
    #else
    scanf("%*[\n] ");
    #endif
}
ip = 0;
for (j = 0; j < mx; j++)
{
    if (isx[j] == 1)
        ip = ip+1;
}
#endif NAG_COLUMN_MAJOR
pdc = my;
#define C(I, J) c[(J-1)*pdc + I-1]
pdp = ip;
#define P(I, J) p[(J-1)*pdp + I-1]
pdt = n;
#define T(I, J) t[(J-1)*pdt + I-1]
pdu = n;
#define U(I, J) u[(J-1)*pdu + I-1]
chw = ip;
#define W(I, J) w[(J-1)*chw + I-1]
pdx = n;
#define X(I, J) x[(J-1)*pdx + I-1]
pdxres = n;
#define XRES(I, J) xres[(J-1)*pdxres + I-1]
py = n;
#define Y(I, J) y[(J-1)*py + I-1]
pycv = maxfac;
#define YCV(I, J) ycv[(J-1)*pycv + I-1]
pyres = n;
#define YRES(I, J) yres[(J-1)*pyres + I-1]
order = Nag_ColMajor;
#else
pdc = maxfac;
#define C(I, J) c[(I-1)*pdc + J-1]
pdp = maxfac;
#define P(I, J) p[(I-1)*pdp + J-1]
pdt = maxfac;
#define T(I, J) t[(I-1)*pdt + J-1]
pdu = maxfac;
#define U(I, J) u[(I-1)*pdu + J-1]
chw = maxfac;
#define W(I, J) w[(I-1)*chw + J-1]
pdx = mx;
#define X(I, J) x[(I-1)*pdx + J-1]
pdxres = ip;
#define XRES(I, J) xres[(I-1)*pdxres + J-1]
py = my;
#define Y(I, J) y[(I-1)*py + J-1]
pycv = my;
#define YCV(I, J) ycv[(I-1)*pycv + J-1]
pyres = my;
#define YRES(I, J) yres[(I-1)*pyres + J-1]
order = Nag_RowMajor;
#endif
if (!(c = NAG_ALLOC(pdc*(order == Nag_RowMajor?my:maxfac), double)) ||
    !(p = NAG_ALLOC(pdp*(order == Nag_RowMajor?ip:maxfac), double)) ||
    !(t = NAG_ALLOC(pdt*(order == Nag_RowMajor?n:maxfac), double)) ||
    !(u = NAG_ALLOC(pdu*(order == Nag_RowMajor?n:maxfac), double)) ||
    !(w = NAG_ALLOC(chw*(order == Nag_RowMajor?ip:maxfac), double)) ||
    !(x = NAG_ALLOC(pdx*(order == Nag_RowMajor?n:mx), double)) ||
    !(xbar = NAG_ALLOC(ip, double)) ||
    !(xcv = NAG_ALLOC(maxfac, double)) ||
    !(xres = NAG_ALLOC(pdxres*(order == Nag_RowMajor?n:ip), double)) ||
    !(ystd = NAG_ALLOC(ip, double)) ||
    !(y = NAG_ALLOC(pdy*(order == Nag_RowMajor?n:my), double)) ||
    !(ybar = NAG_ALLOC(my, double)) ||
    !(ycv = NAG_ALLOC(pdycv*(order == Nag_RowMajor?maxfac:my),
        double)) ||
    !(ystd = NAG_ALLOC(my, double)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
maxit = 200;
tau = 1.00e-4;
/* Read data values.*/
for (i = 1; i <= n; i++)
{
    for (j = 1; j <= mx; j++)
        scanf_s("%lf ", &X(i, j));
    for (j = 1; j <= my; j++)
        scanf_s("%lf ", &Y(i, j));
}
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif
/* Fit a PLS model.*/
/* nag_pls_orth_scores_wold (g02lbc)
 * Partial least-squares
 */
nag_pls_orth_scores_wold(order, n, mx, x, pdx, isx, ip, my, y, pdy, xbar,
    ybar, iscale, xstd, ystd, maxfac, maxit, tau,
    xres, pdxres, yres, pdyres, w, pdw, p, pdp, t,
    pdt, c, pdc, u, pdu, xcv, ycv, pdycv, &fail);
if (fail.code != NE_NOERROR)
{
    fprintf(stderr, "Error from nag_pls_orth_scores_wold (g02lbc).
        %s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* nag_gen_real_mat_print (x04cac)
 * Print real general matrix (easy-to-use)
 */
fflush(stdout);
nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, ip,
    maxfac, p, pdp, "x-loadings, P", 0, &fail);
if (fail.code != NE_NOERROR)
{
    fprintf(stderr, "Error from nag_gen_real_mat_print (x04cac).
        %s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* nag_gen_real_mat_print (x04cac)
 * Print real general matrix (easy-to-use)
 */
fflush(stdout);
nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n,
    maxfac, t, pdt, "x-scores, T", 0, &fail);
if (fail.code != NE_NOERROR)
{
    fprintf(stderr, "Error from nag_gen_real_mat_print (x04cac).
        %s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* nag_gen_real_mat_print (x04cac)
 * Print real general matrix (easy-to-use)
flush(stdout);
nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, my, maxfac, c, pdc, "y-loadings, C", 0, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_gen_real_mat_print (x04cac).\n\n%s\n", fail.message);
    exit_status = 1;
goto END;
}

flush(stdout);
nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, maxfac, u, pdu, "y-scores, U", 0, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_gen_real_mat_print (x04cac).\n\n%s\n", fail.message);
    exit_status = 1;
goto END;
}
printf("\n");
printf("%s\n", "Explained Variance");
printf("\%12s %21s\n", "Model effects", "Dependent variable(s)");
for (i = 1; i <= maxfac; i++)
{
    printf("%12.6f", xcv[i-1]);
    for (j = 1; j <= my; j++)
        printf(" %12.6f%s", YCV(i, j), j%9?' ':"
");
    printf("\n");
}
END:
NAG_FREE(c);
NAG_FREE(p);
NAG_FREE(t);
NAG_FREE(u);
NAG_FREE(w);
NAG_FREE(x);
NAG_FREE(xbar);
NAG_FREE(xcv);
NAG_FREE(xres);
NAG_FREE(xstd);
NAG_FREE(y);
NAG_FREE(ybar);
NAG_FREE(ycv);
NAG_FREE(yres);
NAG_FREE(ystd);
NAG_FREE(isx);
return exit_status;

10.2 Program Data

15 15 1 Nag_PredStdScale 4 : n, mx, my, iscale, maxfac
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 : isx
-2.6931 -2.5271 -1.2871 3.0777 0.3891 -0.0701 1.9607 -1.6324 0.5746 0.0744 -1.7333 0.0902 2.8369 1.4092 -3.1398 0.51
10.3 Program Results

nag_pls_orth_scores_wold (g02lbc) Example Program Results

x-loadings, P

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.6708</td>
<td>-1.0047</td>
<td>0.6505</td>
<td>0.6169</td>
</tr>
<tr>
<td>2</td>
<td>0.4943</td>
<td>0.1355</td>
<td>-0.9010</td>
<td>-0.2388</td>
</tr>
<tr>
<td>3</td>
<td>-0.4167</td>
<td>-1.9983</td>
<td>-0.5538</td>
<td>0.8474</td>
</tr>
<tr>
<td>4</td>
<td>0.3930</td>
<td>1.2441</td>
<td>-0.6967</td>
<td>-0.4336</td>
</tr>
<tr>
<td>5</td>
<td>0.3267</td>
<td>0.5838</td>
<td>-1.4088</td>
<td>-0.6323</td>
</tr>
<tr>
<td>6</td>
<td>0.0145</td>
<td>0.9607</td>
<td>1.6594</td>
<td>0.5361</td>
</tr>
<tr>
<td>7</td>
<td>-2.4471</td>
<td>0.3532</td>
<td>-1.1321</td>
<td>-1.3554</td>
</tr>
<tr>
<td>8</td>
<td>3.5198</td>
<td>0.6005</td>
<td>0.2191</td>
<td>0.0380</td>
</tr>
<tr>
<td>9</td>
<td>1.0973</td>
<td>2.0635</td>
<td>-0.4074</td>
<td>-0.3522</td>
</tr>
<tr>
<td>10</td>
<td>-2.4466</td>
<td>2.5640</td>
<td>-0.4806</td>
<td>0.3819</td>
</tr>
<tr>
<td>11</td>
<td>2.2732</td>
<td>-1.3110</td>
<td>-0.7686</td>
<td>-1.8959</td>
</tr>
<tr>
<td>12</td>
<td>-1.7987</td>
<td>2.4088</td>
<td>-0.9475</td>
<td>-0.4727</td>
</tr>
<tr>
<td>13</td>
<td>0.3629</td>
<td>0.2241</td>
<td>2.6332</td>
<td>2.3739</td>
</tr>
<tr>
<td>14</td>
<td>0.3629</td>
<td>0.2241</td>
<td>-2.6332</td>
<td>2.3739</td>
</tr>
<tr>
<td>15</td>
<td>-0.3629</td>
<td>-0.2241</td>
<td>2.6332</td>
<td>-2.3739</td>
</tr>
</tbody>
</table>

x-scores, T

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.1896</td>
<td>0.3898</td>
<td>-0.2502</td>
<td>-0.2479</td>
</tr>
<tr>
<td>2</td>
<td>0.0201</td>
<td>-0.0013</td>
<td>-0.1726</td>
<td>-0.2042</td>
</tr>
<tr>
<td>3</td>
<td>-0.1889</td>
<td>0.3141</td>
<td>-0.1727</td>
<td>-0.1350</td>
</tr>
<tr>
<td>4</td>
<td>0.0210</td>
<td>-0.0773</td>
<td>-0.0950</td>
<td>-0.0912</td>
</tr>
<tr>
<td>5</td>
<td>-0.0090</td>
<td>-0.2649</td>
<td>-0.4195</td>
<td>-0.1327</td>
</tr>
<tr>
<td>6</td>
<td>0.5479</td>
<td>0.2843</td>
<td>0.1914</td>
<td>0.2727</td>
</tr>
<tr>
<td>7</td>
<td>-0.0937</td>
<td>-0.0579</td>
<td>0.6799</td>
<td>-0.6129</td>
</tr>
<tr>
<td>8</td>
<td>0.2500</td>
<td>0.2033</td>
<td>-0.1046</td>
<td>-0.1014</td>
</tr>
<tr>
<td>9</td>
<td>-1.1005</td>
<td>-0.2992</td>
<td>0.2131</td>
<td>0.1223</td>
</tr>
<tr>
<td>10</td>
<td>-0.1810</td>
<td>-0.4427</td>
<td>0.0559</td>
<td>0.2114</td>
</tr>
<tr>
<td>11</td>
<td>0.0497</td>
<td>-0.0762</td>
<td>-0.1526</td>
<td>-0.0771</td>
</tr>
<tr>
<td>12</td>
<td>0.0173</td>
<td>-0.2517</td>
<td>-0.2104</td>
<td>0.1044</td>
</tr>
<tr>
<td>13</td>
<td>-0.6002</td>
<td>0.3596</td>
<td>0.1876</td>
<td>0.4812</td>
</tr>
<tr>
<td>14</td>
<td>0.3796</td>
<td>0.1338</td>
<td>0.1410</td>
<td>0.1999</td>
</tr>
<tr>
<td>15</td>
<td>0.0773</td>
<td>-0.2139</td>
<td>0.1085</td>
<td>0.2106</td>
</tr>
</tbody>
</table>

y-loadings, C

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.5425</td>
<td>1.0475</td>
<td>0.2548</td>
<td>0.1866</td>
</tr>
</tbody>
</table>

y-scores, U

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1.7670</td>
<td>0.1812</td>
<td>-0.0600</td>
<td>-0.0320</td>
</tr>
<tr>
<td>2</td>
<td>-0.6724</td>
<td>-0.2735</td>
<td>-0.0662</td>
<td>-0.0402</td>
</tr>
<tr>
<td>3</td>
<td>-0.9852</td>
<td>0.4097</td>
<td>0.0158</td>
<td>0.0198</td>
</tr>
<tr>
<td>4</td>
<td>0.2267</td>
<td>-0.0107</td>
<td>0.0180</td>
<td>0.0177</td>
</tr>
<tr>
<td>5</td>
<td>-1.3370</td>
<td>-0.3619</td>
<td>-0.0173</td>
<td>0.0073</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>6</td>
<td>8.9056</td>
<td>0.6000</td>
<td>0.0701</td>
<td>0.0422</td>
</tr>
<tr>
<td>7</td>
<td>-1.0634</td>
<td>0.0332</td>
<td>0.0235</td>
<td>-0.0151</td>
</tr>
<tr>
<td>8</td>
<td>4.2143</td>
<td>0.3184</td>
<td>0.0232</td>
<td>0.0219</td>
</tr>
<tr>
<td>9</td>
<td>-2.1580</td>
<td>-0.2652</td>
<td>0.0153</td>
<td>0.0011</td>
</tr>
<tr>
<td>10</td>
<td>-3.7999</td>
<td>-0.4520</td>
<td>0.0082</td>
<td>0.0034</td>
</tr>
<tr>
<td>11</td>
<td>-0.2033</td>
<td>-0.2446</td>
<td>-0.0392</td>
<td>-0.0214</td>
</tr>
<tr>
<td>12</td>
<td>-0.5942</td>
<td>-0.2398</td>
<td>0.0089</td>
<td>0.0165</td>
</tr>
<tr>
<td>13</td>
<td>-5.6764</td>
<td>0.5487</td>
<td>0.0375</td>
<td>0.0185</td>
</tr>
<tr>
<td>14</td>
<td>4.3707</td>
<td>-0.1161</td>
<td>-0.0639</td>
<td>-0.0535</td>
</tr>
<tr>
<td>15</td>
<td>0.5395</td>
<td>-0.1274</td>
<td>0.0261</td>
<td>0.0139</td>
</tr>
</tbody>
</table>

Explained Variance

<table>
<thead>
<tr>
<th>Model effects</th>
<th>Dependent variable(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16.902124</td>
<td>89.638060</td>
</tr>
<tr>
<td>29.674338</td>
<td>97.476270</td>
</tr>
<tr>
<td>44.332404</td>
<td>97.939839</td>
</tr>
<tr>
<td>56.172041</td>
<td>98.188474</td>
</tr>
</tbody>
</table>

*Mark 25 g02lbc.15 (last)*