This is a WinBUGS program for the artificial example in Chapter 4, Section 4.5.

Model: Standard Structural Equation Model
Data Set Name: YO.dat
Sample Size: N=300

(A) Codes for specification of the model using text-based BUGS language

Click (using the left mouse button) **File** on the tool bar once.

Highlight the **New** option and click once.

Input the following format in the opened blank window.

```
model {
    for(i in 1:N){
        #measurement equation model
        for(j in 1:P){
            y[i,j]~dnorm(mu[i,j],psi[j])
            ephat[i,j]<-y[i,j]-mu[i,j]
        }
        mu[i,1]<-eta[i]+alp[1]
        mu[i,2]<-lam[1]*eta[i]+alp[2]
        mu[i,3]<-lam[2]*eta[i]+alp[3]
        mu[i,4]<-xi[i,1]+alp[4]
        mu[i,5]<-lam[3]*xi[i,1]+alp[5]
        mu[i,6]<-lam[4]*xi[i,1]+alp[6]
        mu[i,7]<-xi[i,2]+alp[7]
        mu[i,8]<-lam[5]*xi[i,2]+alp[8]
        mu[i,9]<-lam[6]*xi[i,2]+alp[9]

        #structural equation model
        xi[i,1:2]~dmnorm(u[1:2],phi[1:2,1:2])
        eta[i]~dnorm(nu[i],psd)
        nu[i]<-gam[1]*xi[i,1]+gam[2]*xi[i,2]
        dthat[i]<-eta[i]-nu[i]
    } #end of i

    #priors on intercepts
    for(j in 1:9){alp[j]~dnorm(0.0, 1.0)}

    #priors on loadings and coefficients
    lam[1]~dnorm(0.8,psi[2])
    lam[2]~dnorm(0.8,psi[3])
    lam[3]~dnorm(0.8,psi[5])
    lam[4]~dnorm(0.8,psi[6])
    lam[5]~dnorm(0.8,psi[8])
    lam[6]~dnorm(0.8,psi[9])
    for(j in 1:2){gam[j]~dnorm(0.5, psd)}
```

```
    #priors on precisions
    for(j in 1:P){
        psi[j]~dgamma(9.0, 4.0)
        sgm[j]<-1/psi[j]
    }
    psd~dgamma(9.0, 4.0)
    sgd<-1/psd
    phi[1:2,1:2]~dwish(R[1:2,1:2], 5)
    phx[1:2,1:2]<-inverse(phi[1:2,1:2])

} #end of model
```

(B) Check model

Click **Model** on the tool bar once.

Highlight the **Specification** option and click once.

Highlight the word *model* at the beginning of the aforementioned written code.

Click **check model** in the Specification Tool window once.

Errors messages will be show in the status bar at the bottom left of the WinBUGS program window, and the cursor will be at the place where the error was found. If there are no errors,
then the message `*model is syntactically correct*' will appear at the bottom left of the WinBUGS program window.

(C) Load data

Continue to input the following format for the data in the codes window (follow the last line of the codes that is given in **(A)**) or in a new blank white window.

Data

```
list(N=300, P=9, u=c(0,0),
    y=structure(
       .Data=c(paste YO.dat here),
        .Dim=c(300,9)),
    R=structure(
        .Data=c(1.0,  0.0,
                  0.0, 1.0),.Dim=c(2,2)))
```

Highlight the word *list* in this format.

Click **load data** in the Specification Tool window once.

When the data have been successfully loaded, the message `*data loaded*' should appear at the bottom left of the WinBUGS program window.

(D) Compile model

By default, the program simulate one chain of observations. To generate three chains with three different sets of initial values, type *3* in the box labeled *num of chains* in the Specification Tool window.

Click **compile** in the Specification Tool window once. The message `*model compiled*' should appear at the bottom left of the WinBUGS program window.

(E) Load initial values

Continue to input the following format in the codes window or a new blank window.

Three different initial values

```
list(alp=c(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0),
     lam=c(0.0, 0.0, 0.0, 0.0, 0.0, 0.0),
     psi=c(1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0),
     psd=1.0,      gam=c(1.0, 1.0),
     phi=structure(
        .Data=c(0.2, 0.1,
                0.1, 0.9),
        .Dim=c(2,2)))

list(alp=c(-2.0, -2.0, -2.0, -2.0, -2.0, -2.0, -2.0, -2.0, -2.0),
     lam=c(-1.0, 0.0, 0.3, 0.6, 0.9, 1.0),
     psi=c(1.5, 1.5, 1.5, 1.5, 1.5, 1.5, 1.5, 1.5, 1.5),
     psd=1.5,      gam=c(-1.0,-1.0),
     phi=structure(
        .Data=c(0.5, 0.2,
                0.2, 0.6),
        .Dim=c(2,2)))

list(alp=c(2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0),
     lam=c(1.0, 0.3, 0.4, 0.5, 1.0, -1.0),
     psi=c(0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3),
     psd=0.3,      gam=c(2.0, 2.0),
     phi=structure(
        .Data=c(0.9, 0.3,
                0.3, 0.8),
        .Dim=c(2,2)))
```

Highlight the word *list* in these formats to input the first initial values.

Click **load inits** in the Specification Tool window once.

Click the **gen inits** button in the Specification Tool window once. This is necessary to generate the nodes *xi*s' in the MCMC algorithm.

If the model is fully initialized after loading the initial values, then a message saying '*initial values loaded: model initialized*' will appear at the bottom left of the WinBUGS program
window. Otherwise, the message '*initial values loaded: model contains uninitialized nodes*' will appear at the bottom left of the WinBUGS program window.

(F) Run simulation and check convergence

Step 1: Input the monitored parameters

Click **Inference** on the tool bar once.

Click **Samples** once. A Sample Monitor Tool window will appear.

Type the node of the parameters (for example, the parameter vector *bb*) to be monitored into the box marked *node* in the Sample Monitor Tool window.

Click **set** in the Sample Monitor Tool window once.

Step 2: Sample observations for the monitored node

Click **Model** on the tool bar once.

Click **Update** option once. A Update Tool window will appear.

Type the total number of MCMC iterations (for example, 5000 or 10000; the default value is 1000) into the box named *updates* in the Update Tool window.

Type the number of refresh (i.e., that are requested how often the screen is refreshed to show how the sampling is proceeding) into the box marked *refresh* in the Update Tool window. The default value is 100.

Type the number of thin (for example, k, i.e., the samples from every k-th iteration can be stored) into the box marked *thin*. The default is 1.

Click **update** in the Update Tool window once. Wait for the program to finish simulating the specified number of iterations.

When finished, the message `*updates took* ***s*' will appear in the bottom left of the WinBUGS program window.

Step 3: Check convergence

Point to the Sample Monitor Tool window in Step 1.

Type the previously monitored node that is given in Step 1 (for example, *bb*) into the box marked *node* in the Sample Monitor Tool window.

The first few thousand iterations (say J) should be treated as burn-in iterations. This is carried out by specifying the number J in the box marked *beg* in the Sample Monitor Tool window. The remaining iteration will be used to obtain the Bayesian statistics.

Click **trace** in the Sample Monitor Tool window once. Plots of the simulated observations against the number of iterations for all of the parameters in *bb* will be shown for checking whether the algorithm has converged.

Step 4: Obtain Bayesian statistics

Click **stats** in the Sample Monitor Tool window once. The results for statistics inference, for example, the mean (the Bayesian estimates), standard deviation, Monte Carlo error, will be given.

To get estimates for omega_i (i.e. node *xi*), repeat the aforementioned steps 1-3 with additional iterations (for example, 4000 or 5000). Note that the output results are displayed
by a column vector
(xi[1,1],xi[1,2],xi[1,3],xi[2,1],xi[2,2],xi[2,3],.....,xi[100,1],xi[100,2],xi[100,3]).

To get the corresponding DIC values, one can do as follows after the program was converged:
Click **Inference** on the tool bar once.
Click **DIC** once. A DIC window will appear, and then click **Set** in this window.
Return to the **update** window, and let the program run some iterations, say 5000 iterations, then go back to the **DIC** window and click **DIC.** A window which contains the DIC value will pop up.