

Problem Set 2

Due date: Tuesday, February 16, in class.

Derive the impulse response functions for the following time series models.

- ARMA(1,1) $y_t = c + \phi y_{t-1} + \varepsilon_t + \theta \varepsilon_{t-1}$
- ARMA(1,2) $y_t = c + \phi y_{t-1} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2}$
- ARMA(2,2) $y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2}$
- ARMA(3,2) $y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \phi_3 y_{t-3} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2}$
- AR(4) $y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \phi_3 y_{t-3} + \phi_4 y_{t-4} + \varepsilon_t$

For the following parameter values, confirm stationarity and invertibility of the AR and MA polynomials respectively, then plot the first 100 impulse responses in Gauss. Label all of your graphs, and turn in your Gauss code, as well as the written derivation of the above impulse response functions.

$$y_t = 2 + 0.9y_{t-1} + \varepsilon_t + 0.5\varepsilon_{t-1}$$

$$y_t = 4 + 0.7y_{t-1} + \varepsilon_t + 0.6\varepsilon_{t-1} + 0.2\varepsilon_{t-2}$$

$$y_t = 8 + 1.34y_{t-1} - 0.71y_{t-2} + \varepsilon_t + 0.7\varepsilon_{t-1} - 0.1\varepsilon_{t-2}$$

$$y_t = 16 + 1.25y_{t-1} - 0.65y_{t-2} + 0.2y_{t-3} + \varepsilon_t + 0.5\varepsilon_{t-1} + 0.4\varepsilon_{t-2}$$

$$y_t = 32 + 2.7607y_{t-1} - 3.8106y_{t-2} + 2.6535y_{t-3} - 0.9238y_{t-4} + \varepsilon_t$$

Gauss Hints

Procedures. It is often easier to create procedures in Gauss, which you can then treat as intrinsic functions in your programs. Consider the following procedure to compute the IRF for an AR(2).

```
@ procedure to generate the IRF for an AR(2) @
```

```
proc(1)=irf_AR2(phi,j); @ 1 denotes the number of outputs from the proc @
```

```
/* phi and j will be specified outside of the procedure and will be treated as global variables within the procedure */
```

```
@ phi is a 2 x 1 vector of AR coefficients, and j is the length of the IRF @
```

```
local i,irf,phi1,phi2;
```

```
/* local variables are only used within the procedure, and they must all be declared */
```

```
phi1=phi[1]; phi2=phi[2];  
irf=zeros(j,1);
```

```
irf[1]=phi1; irf[2]=phi1^2+phi2; @ initial conditions @
```

```
@ iterate on the Yule-Walker equations @
```

```
i=3; do until i>j;  
irf[i]=phi1*irf[i-1]+phi2*irf[i-2];  
i=i+1; endo;
```

```
retp(irf);  
endp;
```

```
/* all procedures must end with a “retp” command, which returns to desired output, and an “endp” command, which ends the procedure */
```

```

/* program to graph the first 100 IRFs for an AR(2) */

new;
library pgraph;
phi=1.1|-0.2;
j=100;

@ check for stationarity @

F=phi'(1~0);

lambda=eig(F); @ eigenvalues of F @
mod1=abs(lambda);

/* equivalently we could solve the roots of the AR polynomial */

c=-phi[2]-phi[1]|1;
z=polyroot(c);
@ this solves (1-phi1*z-phi2*z^2=0) for z @

mod2=abs(z); @ these should be the inverses of the mod of lambda @

irf=1lirf_AR2(phi,j);
xy(seqa(0,1,j+1),irf);
end;

proc(1)=irf_AR2(phi,j);

local i,irf,phi1,phi2;

phi1=phi[1]; phi2=phi[2];
irf=zeros(j,1);

irf[1]=phi1; irf[2]=phi1^2+phi2;

i=3; do until i>j;
irf[i]=phi1*irf[i-1]+phi2*irf[i-2];
i=i+1; endo;

retp(irf);
endp;

```