






Corres código Dynare. Luego de que funciona y obtienes las funciones de impulso respuesta se escribe el siguiente código en el Command Window :

```
save oo_.irfs
```

```
>> save oo_.irfs
```

Si todo salió correctamente, no obtenemos ninguna respuesta en la ventana de comando, pero si se genera un archivo llamado oo\_.irfs en la carpeta del código de Dynare. Se genera sin extensión, por lo que agregamos la extensión que le corresponde que es .mat (Si uno tuviese Matlab el archivo se genera automáticamente). La carpeta debería verse algo así:

 +Tebaldi	1/21/2025 10:36 PM
 Tebaldi	1/21/2025 10:36 PM
 oo_.irfs.mat	1/28/2025 7:36 PM
 Tebaldi.log	1/5/2025 12:15 PM
 Tebaldi.mod	1/28/2025 7:17 PM

Vamos a python para abrir el archivo .mat para eso ocupamos las dependencias Numpy y Scipy.

```
import scipy.io  
  
irfs_data = scipy.io.loadmat('oo_.irfs.mat')
```

Este es un diccionario gigante con un montón de cosas. Las funciones de impulso respuesta se encuentran en la clave oo\_ , por lo tanto:

```
irfs = irfs_data['oo_']
```

Lo cual genera un diccionario también gigante, Matlab Please stop! I just wanted to plot the irfs not to get the code to decipher how to solve the Turing Test. Para acceder al array que contiene a las funciones de impulso respuesta entonces:

```
IRF = irfs[0][0][-2]
```

Ahora separamos cada una por su variable, generando un nuevo dicc ordenado con su clave:

```
irf_variables = {var: np.array(IRF[var][0]) for var in IRF.dtype.names}
```

Y luego Podemos guardar los datos de cada variable endógena accediendo a cada elemento, por ejemplo:

```
Y_e = irf_variables['Y_e'][0][0]
```

El doble [0] es porque es un array dentro de otro dentro de otro, si uno no lo pone entonces no ValueError al graficar.

```
[29]: import scipy.io

irfs_data = scipy.io.loadmat('Tebaldi\oo_.irfs.mat')
print(irfs_data.keys())

dict_keys(['__header__', '__version__', '__globals__', 'M_', 'alpha', 'ar',
'deltaA', 'estim_params_', 'estimation_info', 'etaC', 'etaL', 'gamma', 'i
```

```
[30]: # Assuming the IRFs are stored under 'oo_.irfs'
irfs = irfs_data['oo_']

# Print the shape of the IRFs array to understand its structure
print(irfs.shape)

(1, 1)
```

```
[31]: import numpy as np
IRF = irfs[0][0][-2]
irf_variables = {var: np.array(IRF[var][0]) for var in IRF.dtype.names}
# Now, each variable is stored in `irf_variables` dictionary

Y_e = irf_variables['Y_e'][0][0]
C_e = irf_variables['C_e'][0][0]
A_e = irf_variables['A_e'][0][0]
LA_e = irf_variables['LA_e'][0][0]
PA_e = irf_variables['PA_e'][0][0]
K_e = irf_variables['K_e'][0][0]
```