Macroeconomia Microfundamentada

Resolvendo Modelos DSGE

Tomás R. Martinez

INSPER

Referências

- Celso José Costa Junior: Cap. 2.
- Mario Solís-Garia: Lec. 6.
- Mario Solís-Garia: Handout 2 (Matlab) e 5 (Dynare).
- Eric Sims: notes on Dynare.

Introdução

- Vimos que a solução teórica do modelo de ciclos reais de negócios é uma sequência de infinitas equações não-lineares.
- O modelo de ciclos reais é o modelo DSGE mais básico e não tem forma fechada. Outros modelos são ainda mais complexos.
- Vamos ver um tipo de solução numérica desses modelos: a solução por log-linearização ao redor do estado estacionário.
- Posteriormente iremos implementar a solução no computador utilizando o **Dynare**, um pacote do **Matlab**.

Exemplo

- Considere uma versão simples do modelo RBC, onde a oferta de trabalho é constante: $n = 1$ e a utilidade é log (esse é o modelo de crescimento neoclássico estocástico).

- As equações de equilíbrio são para todo $t = 0, 1, ...\infty$:

$$\frac{1}{c_t} = \beta \mathbb{E}_t[(1 + \alpha z_{t+1} k_{t+1}^{\alpha-1} - \delta)\frac{1}{c_{t+1}}]$$
$$z_t k_t^{\alpha} = c_t + k_{t+1} - (1 - \delta)k_t$$
$$\ln(z_t) = \rho \ln(z_{t-1}) + \varepsilon_t$$

- E as as variáveis no estado estacionário, $\bar{k}$ e $\bar{c}$, são funções dos parâmetros:

$$\bar{k} = \left[\frac{\alpha \bar{z}}{1/\beta - (1 - \delta)}\right]^{\frac{1}{1-\alpha}} \qquad \text{e} \qquad \bar{c} = \bar{z}\bar{k}^{\alpha} - \delta\bar{k}$$

Policy Functions

- A solução do sistema é uma sequência de capital e consumo, $\{c_t, k_{t+1}\}_{t=0}^{\infty}$, que eventualmente converge para o estado estacionário (se não houver choques).
- Encontrar a sequência infinita é muito complicado já que o sistema não tem solução analítica.
- Contudo, como o sistema é "infinito", em todos os períodos o problema é igual. A única coisa que muda é o capital da economia k_t e o choque z_t.
- Logo, a solução para o consumo c_t e para o capital do próximo período (a poupança), k_{t+1}, são funções dessas variáveis:

$$k_{t+1} = g_k(k_t, z_t) \qquad \text{e} \qquad c_t = g_c(k_t, z_t)$$

- Essas funções são chamadas de **policy functions**. As variáveis k_t e z_t são conhecidas como **state variables**. As outras variáveis do modelo são conhecidas como **control variable** ou **jump variables**.

Log-linearização

- Podemos encontrar as **policy function** do sistema de diferente maneiras.
- Uma maneira prática é log-linearizar o sistema ao redor do estado estacionário.
- A linearização é uma equação aproximada das equações não-lineares.
- A aproximação é válida, já que o sistema sempre retorna para o estado estacionário após choques temporários.
- Caso o choque seja muito grande e a dinâmica leve a muito longe do estado estacionário, a aproximação pode ser ruim.

Log-linearização

- Suponha uma variável agregada x_t, onde $\bar{x}$ é o seu valor no estado estacionário.

$$\tilde{x}_t = \log\left(\frac{x_t}{\bar{x}}\right) = \underbrace{\log(x_t) - \log(\bar{x})}_{\text{desvios \% do estado estacionário}} \approx \frac{x_t - \bar{x}}{\bar{x}}$$

- Também podemos re-escrever as variáveis dessa forma: $x_t = \bar{x}e^{\tilde{x}_t}$ e $e^{\tilde{x}_t} \approx 1 + \tilde{x}_t$.

- O nosso objetivo é escrever as equações de equlíbrio do modelo log-linearizadas.

Log-linearização

- É possível linearizar as equações do modelo com papel e caneta. Mas esse é um processo tedioso e não muito prático. É mais conveniente deixar o computador fazer os cálculos.
- Caso você queira entender mais como fazer isso veja as referências do Solis-Garcia e aqui.
- Após a log-linearização, o sistema de equações (lineares) de equilíbrio fica:

$$
\begin{aligned}
-\tilde{c}_t &= \mathbb{E}_t[-\tilde{c}_{t+1} + \tilde{r}_{t+1}] \\
\bar{r}\tilde{r}_{t+1} &= \alpha \bar{z}\bar{k}^{\alpha-1}(\tilde{z}_{t+1} + (\alpha-1)\tilde{k}_{t+1}) \\
[\alpha \bar{z}\bar{k}^{\alpha} + (1-\delta)\bar{k}]\tilde{k}_t &= \bar{c}\tilde{c}_t + \bar{k}k_{t+1} \\
\tilde{z}_t &= \rho \tilde{z}_{t-1} + \varepsilon_t
\end{aligned}
$$

- Não se preocupe em saber como chegar nessas equações. A ideia geral é que podemos aproximar um sistema não-linear por um linear.

Log-linearização e Policy Functions

- Após a linearização, podemos encontrar as policy functions.
- Como fazer isso? Sabemos que as policy function são funcões das variáveis estado:

$$\begin{aligned}\tilde{k}_{t+1} &= \Psi_{kk}\tilde{k}_t + \Psi_{kz}\tilde{z}_t \\ \tilde{c}_t &= \Psi_{ck}\tilde{k}_t + \Psi_{cz}\tilde{z}_t\end{aligned}$$

- Eu posso utilizar as equações acima nas equações de equilíbrio e encontrar os valores de parâmetros Ψ's.
- Uma vez que temos os Ψ's, conseguimos saber como o modelo se comporta, simular funções impulso-resposta e etc.
- Problema é que encontrar os Ψ's no papel e caneta é um processo difícil e trabalhoso. Vamos usar o computador!

Simulando Modelos DSGE no Dynare

Matlab e Dynare

- Vamos simular os modelos no Matlab, utilizando o pacote Dynare.
- O que é o **Matlab**? MATrix LABoratory.
  - É um software utilizado para computação numérica, especialmente em operações matriciais.
  - Milhões de funções, além de ser fácil criar outros.
  - É possível importar dados, rodar regressões, modelos econométricos e etc. Bastante usado por quem estuda macroeconomia.
- É um software pago, mas o Insper tem licenças acadêmicas.
- Faça o download aqui, use o seu email do Insper para conseguir a licença. Qualquer dúvida fale com o helpdesk.

Matlab

- Para uma introdução rápida sobre como usar o Matlab veja notas do Solis-Garcia.
- Muitas outras notas introdutórias sobre o programa na internet.
- Se você tiver experiência com outras linguagens (R, Python, etc) irá encontrar muitos elementos em comum:
  - Você pode escrever seu código em um script e depois rodar o script.
  - Pode escrever diretamente na janela de comando.
  - Tem que definir o diretório, as variáveis ficam salvas na memória, etc.

Matlab

- Na visualização básica do Matlab, você vê algumas janelas:
  - Command Window: A janela que você avalia digita comandos e expressões que são executadas imediatamente.
  - Current Folder e Current Directory: Seu diretório de trabalho. Aqui você vê os arquivos.
  - Workspace window: Aqui estão as variáveis sendo usadas na memória.
  - Editor: Esse é seu script (ou um arquivo que você pode escrever seu script).
- Os scripts são salvos em *.m-files*. Existem dois tipos de m-files. Scripts usuais que podem ser executados na janela de comando ou scripts que definem funções.
- Você pode salvar suas variáveis da workspace window em uma *.mat-file*.
- Abra o script `MatlabBasics.m` para ver exemplos de comandos.

Dynare

- Resolver modelos DSGE por linearização é uma tarefa tediosa.
- Mas existem softwares especializados que tornam a tarefa mais fácil. O mais usado deles: **Dynare**.
- Dynare é um pacote do Matlab que resolve e estima modelos DSGE. Também está disponível para Julia.
  - Foi desenvolvido por um grupo de pesquisadores do CEMPREMAP na França.
  - Hoje é bastante usados por pesquisadores em geral, principalmente em Bancos Centrais ao redor do mundo.
- Antes de tentar rodar algum modelo mais complicado: leia as minhas notas de aula com cuidado, notas do Solis-Garcia, e abra e rode o programa dos modelos mais simples!
  - Para dúvidas mais profundas olhe o manual do Dynare. É bem detalhado!

Instalando o Dynare

- Primeiro instale o Matlab. Depois baixe e instale a versão mais recente compatível com o seu computador pelo http://www.dynare.org.
- Um arquivo do Dynare é identificado por uma extensão `.mod`.
- O `.mod` é um arquivo que contem toda a informção do seu modelo e os comandos para solucioná-los. É um script em "plain text", e você pode criar um pelo Notepad ou pelo próprio Matlab.
- Para rodar um script `.mod` no Matlab é necessário preparar o Dynare no seu Matlab.

Rodando uma .mod-file no Matlab

- Primeiro você precisa fazer com que o Matlab reconheça o diretório do Dynare. Escreva o seguinte comando na janela de comandos do Matlab:
  - `addpath /Applications/Dynare/6.0/matlab` (se estiver usando um Mac)
  - `addpath c:\dynare\6.0\matlab` (se estiver usando um Windows)
  - Note que isso pode ser diferente dependendo da versão que você instalou do dynare, do diretório utilizado e etc. Veja no seu computador exatamente onde o dynare foi instalado para não cometer nenhum erro.

- Depois mude o diretório do Matlab para a pasta onde estão localizadas suas `.mod` files com o comando "cd":
  - `cd('c:\minha pasta')`

- Finalmente, rode o seu modelo escrevendo na janela de comandos:
  - `dynare simple_model1.mod`
  - Se não houver erros deve aparecer bastante informação na janela de comando do Matlab.

Escrevendo uma .mod-file

- Um `.mod` file consiste de cinco partes:
  - **Preamble**: define as variáveis endógenas do modelo e os parâmetros.
  - **Model**: as equações do modelo.
  - **Steady state**: valores para o estado estacionário.
  - **Shocks**: aqui você define os choques do modelo que você deseja simular.
  - **Computation**: aqui você diz como quer simular o modelo e reportar os resultados.
- Alguns detalhes:
  - Comandos terminam com ponto e vírgula: ;
  - Para escrever comentários utilize: \\ ou %.
  - Alguns blocos de comando terminam com `end;`

Preamble

- Como exemplo vamos resolver o modelo apresentado anteriormente:

$$\frac{1}{c_t} = \beta \mathbb{E}_t[(1 + r_{t+1} - \delta)\frac{1}{c_{t+1}}]$$
$$z_t k_t^{\alpha} = c_t + k_{t+1} - (1 - \delta)k_t$$
$$r_t = \alpha z_t k_t^{\alpha - 1}$$
$$\ln(z_t) = \rho \ln(z_{t-1}) + \varepsilon_t$$

  - Variáveis endógenas: $c,\ k,\ r,\ z$.
  - Parâmetros: $\beta,\ \delta,\ \alpha,\ \rho$.

- Esse modelo está dentro do: `simple_model1.mod`

Preamble: variáveis

- No preamble você tem que definir as variáveis endógenas e exógenas, os parâmetros e dar valores para os parâmetros.
- Variáveis endógenas são definidas depois de `var` e as exógenas depois de `varexo`:

```
 9   // Endogenous variables of the model
10   var z c k r;
11
12   // Exogenous shock
13   varexo ez ;
14
```

- A variável exógena é o choque no processo estocástico: ε_t (e não z_t).

Preamble: parâmetros

- Os parâmetros são escritos depois de `parameters`. No nosso modelo: β, δ, α, ρ.

```
15  // Parameters of the model
16  parameters beta delta alpha rho ;
17
18  // Calibration of parameters
19  beta=0.99;
20  alpha=1/3;
21  delta=0.02;
22  rho =0.979;
23
```

- Em modelos linearizados podemos definir as variáveis no estado estacionário como parâmetros.

Preamble: Equações do modelo

- Precisamos colocar as equações do modelo depois de `model;` e terminar com `end;`.

$$\frac{1}{c_t} = \beta \mathbb{E}_t[(1 + r_{t+1} - \delta)\frac{1}{c_{t+1}}]$$
$$z_t k_t^{\alpha} = c_t + k_{t+1} - (1 - \delta)k_t$$
$$r_t = \alpha z_t k_t^{\alpha - 1}$$
$$\ln(z_t) = \rho \ln(z_{t-1}) + \varepsilon_t$$

```
// Model equations:
model;
1/exp(c) = 1/ exp(c(1)) * beta*(1 - delta + r(1));                    // Euler equation
exp(z)*exp(k(-1))^alpha = exp(c) + exp(k) -(1-delta)*exp(k(-1));      // Market clearing
r = alpha*exp(z)*exp(k(-1))^(alpha-1);                                // Interest rate
z=rho*z(-1) + ez;                                                     // Shock process
end;
```

Model: Equações do modelo

- A grande vantagem do Dynare é que você não precisa escrever o modelo linearizado (mas se quiser você pode fazer isso). Basta escrever as equações do modelo.
- Note que definimos as variáveis com $\exp(x)$: intuitivamente, se $\hat{c} = \ln(c) \Rightarrow \exp(\hat{c}) = c$.
- Ajuda a interpretar o modelo: as funções impulso-resposta são interpretadas com mudanças em % e não mudanças em níveis.
  - Alternativamente você poderia definir as variáveis em log dentro do próprio código. Veja `simple_model3.mod`.
- O timing do Dynare:
  - $r(1) = r_{t+1}$, $r = r_t$ e $r(-1) = r_{t-1}$.
  - A variável recebe o "timing" quando ela é decidida de fato. Note que no modelo temos que k_t, mas o capital é decidido período anterior, logo no Dynare escrevemos $k(-1)$.

Steady State

- O Dynare computa os valores no estado estacionário do modelo numericamente.
- Para isso basta escrever `steady;`.
- Opcionalmente podemos colocar um chute inicial para o algoritmo numérico do Dynare via `initval;`. Se o modelo for complicado isso pode acelerar o processo.

```
37  initval ; \\ optional
38  z=1; k=10; c = 1.5; r = 0.05;
39  end ;
40
41  steady;
```

- Se você não colocar `steady;` o dynare irá utilizar o `initval;` como condição inicial da simulação.

Steady State

- Se você conseguir calcular o estado estacionário do modelo na mão, você pode colocar diretamente no Dynare e poupar tempo.
- No nosso caso seria algo do tipo:

```
46  steady_state_model ;
47  k = (alpha/(1/beta - 1 + delta))^(1/(1-alpha));
48  c = (alpha/(1/beta - 1 + delta))^(alpha/(1-alpha))  - delta*(alpha/(1/beta - 1 + delta))^(1/(1-alpha));
49  end ;
50  steady;
```

Steady State

- Após colocarmos o steady state e as equações do modelo podemos fazer alguns diagnósticos do modelo:
  - `check;` $\Rightarrow$ irá verificar se as condições de estabilidade são satisfeitas.
  - `model_diagnostics;` $\Rightarrow$ irá verificar se tem algum problema no modelo apresentado.
  - `model_info;` $\Rightarrow$ irá te dar algumas informações sobre o modelo, como quantidade de choques, variáveis de estado e etc.
- Tudo isso é opcional, mas altamente recomendável.

Digressão: condições de estabilidade

- O modelo é dinâmico: tem equações de diferenças com variáveis "forward looking", c_{t+1}, e "backward looking", k_{t+1}.
- Dependendo dos parâmetros o modelo pode não ter solução estável.
- **Exemplo:** $x_t = \rho x_{t-1} + \varepsilon_t$.
  - Para o AR(1) ter solução estável precisamos de $-1 < \rho < 1$.
  - Caso $\rho > 1$ um choque em ε_t faz com que x_t "exploda" e não retorne ao estado estacionário.
- Essas são chamadas de **blanchard-kahn conditions**.
- Quando o dynare escreve: `The rank condition is verified.` é porque as condições foram satisfeitas.

Shocks

- Aqui escrevemos as propriedades do choque $\varepsilon \sim N(0, \sigma_z)$ (lembre-se: $z_t = \rho z_{t-1} + \varepsilon_t$).
- Basicamente é definir a variância do choque (e a correlação no caso houver mais de 1 choque).
  - `var` define a variável de choque.
  - `stderr` o standard error do choque.

```
52  shocks;
53  var ez;
54  stderr 0.01;
55  end;
```

- Para computar as funções impulso resposta o dynare dá o choque de um desvio padrão. Eu normalmente escrevo 0.01 porque podemos interpretar como um choque de 1% no modelo loglinearizado.

Simulation

- A parte de simulação é auto explicativa. Para computar os momentos e funções impulso-resposta do modelo escreva: `stoch_simul(...)`.
- Muitas opções! Veja o manual para detalhes.
- Por exemplo: `stoch_simul(order=1, irf = 200)` indica uma aproximação de Taylor do modelo de ordem 1 e que simulamos a função impulso resposta por 200 períodos.
  - Outras opções: `noprint`, `hpfilter`, `ar`, `nocorr`...
- Um truque interessante é que podemos chamar `stoch_simul(...)` quantas vezes quizermos e mudarmos os parâmetros por exemplo.

Output

- Ok, você escreveu o comando: `dynare simple_model1.mod`. Muitas coisas apareceram na janela de comando do matlab, como interpretar?
  - (i) Preprocessor output.
  - (ii) Steady state results.
  - (iii) Eigenvalues and model summary.
  - (iv) Matrix of covariance of exogenous shocks.
  - (v) Policy and transition functions.
  - (vi) Theoretical moments.
  - (vii) Matrix of correlations.
  - (viii) Coefficients of autocorrelation.
  - (ix) Impulse-response function graphs.

Preprocessor output

- Se o modelo foi especificado corretamente na `.mod-file`, irá aparecer um bocado de texto:

```
Starting Dynare (version 6.0).
Calling Dynare with arguments: none
Starting preprocessing of the model file ...
Found 4 equation(s).
Evaluating expressions...
Computing static model derivatives (order 1).
Normalizing the static model...
Finding the optimal block decomposition of the static model...
2 block(s) found:
  1 recursive block(s) and 1 simultaneous block(s).
  the largest simultaneous block has 3 equation(s)
                                  and 3 feedback variable(s).
Computing dynamic model derivatives (order 1).
Normalizing the dynamic model...
Finding the optimal block decomposition of the dynamic model...
2 block(s) found:
  1 recursive block(s) and 1 simultaneous block(s).
  the largest simultaneous block has 3 equation(s)
                                  and 3 feedback variable(s).
Preprocessing completed.
Preprocessing time: 0h00m00s.
```

- Se tiver algum erro na sua `.mod-file` (tipo um ; fora do lugar, alguma variável não declarada, etc), o dynare vai te avisar aqui.

Steady state results

- Aqui estarão os valores das variáveis no estado estacionário.

```
STEADY-STATE RESULTS:

z                0
c                0.951936
k                3.60688
r                0.030101
```

- Novamente, se houver algum erro ou o algoritmo do Dynare não conseguir encontrar um estado estacionário ele te avisará por aqui.

- Logo após iremos ter **Eigenvalues and model summary**. Não tem muita interpretação sobre essa parte. Dando tudo certo você não precisará se preocupar, mas se houver algum erro provavelmente seu modelo não tem uma solução estável (parâmetros explosivos???).

Matrix of covariance of exogenous shocks

- Com um só choque essa parte não será muito interessante.
- Mas se seu modelo tiver muitos choques exógenos, a matriz pode ser interessante para entender os links entre os choques.

```
MATRIX OF COVARIANCE OF EXOGENOUS SHOCKS
Variables           ez
ez            0.000100
```

Policy and transition functions

- As policy functions são uma parte importante porque te permite a simular o modelo fora do dynare, fazer diagnóstico e entender como o modelo funciona de uma maneira geral.

```
POLICY AND TRANSITION FUNCTIONS
                                      z             c             k             r
Constant                              0      0.951936      3.606875      0.030101
z(-1)                          0.979000      0.406074      0.059858      0.029469
k(-1)                                 0      0.599747      0.967937     -0.020067
ez                             1.000000      0.414784      0.061142      0.030101
```

- Precisamos ter certo cuidado para interpretar essas funções já que as funções estão em desvios do estado estado estacionário.

Policy and transition functions

- Por exemplo, o consumo:

$$c_t = 0.951 + 0.406\hat{k}_{t-1} + 0.599\hat{z}_{t-1} + 0.414\varepsilon_t,$$

  onde $\hat{k}_t = k_t - k_{ss}$ é o desvio do estado estacionário. Como escrevemos o modelo com o truque $\exp$, os desvios são em % (i.e., k_t na verdade é $\log(k_t)$).

- Note que quando $\hat{k}_{t-1} = \hat{z}_{t-1} = \varepsilon_t = 0$ o valor de c_t é o estado estacionário.

- Em $t = 0$, no momento do choque $\varepsilon_t = 0.01$, e podemos calcular c_t. Com isso podemos calcular $\hat{c}_t = c_t - c_{ss}$ e assim por diante.

- É basicamente isso a função impulso resposta faz.

Theoretical moments

- O Dynare computa os momentos (média, desvio padrão, etc) teóricos das variáveis endógenas do modelo utilizando as policy functions:

```
THEORETICAL MOMENTS
VARIABLE          MEAN  STD. DEV.   VARIANCE
z               0.0000     0.0491     0.0024
c               0.9519     0.0607     0.0037
k               3.6069     0.0728     0.0053
r               0.0301     0.0010     0.0000
```

- Caso você queira os momentos simulados, o dynare pode gerar dados artificiais com o modelo. Basta escrever `stoch_simul(..., periods = 150000, drop = 50000)`.
  - Estamos gerando 150k dados, dropando os 50k primeiros e calculando as estatísticas com os 100k de dados que sobraram.

Matrix of correlations

- Basicamente as correlações das variáveis endógenas do modelo.
- Muito útil para entender quais variáveis são pro-cíclicas ou anti-cíclicas.

```
MATRIX OF CORRELATIONS
Variables           z         c         k         r
z              1.0000    0.8885    0.7864    0.3531
c              0.8885    1.0000    0.9822   -0.1156
k              0.7864    0.9822    1.0000   -0.3003
r              0.3531   -0.1156   -0.3003    1.0000
```

Coefficients of autocorrelation

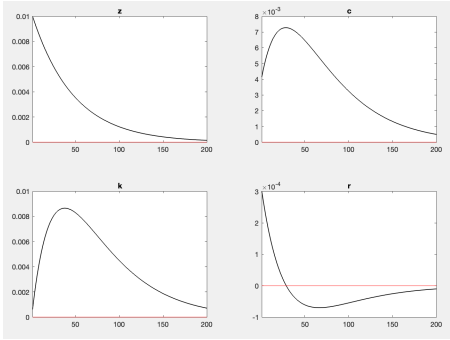
- De maneira similar podemos calcular a matriz de autocorrelação dos dados do modelo.

```
COEFFICIENTS OF AUTOCORRELATION
Order          1       2       3       4       5
z         0.9790  0.9584  0.9383  0.9186  0.8993
c         0.9975  0.9946  0.9910  0.9871  0.9826
k         0.9997  0.9987  0.9970  0.9948  0.9921
r         0.9524  0.9067  0.8627  0.8205  0.7799
```

- Útil para checar com os dados reais e comparar seu modelo!

Gráficos da função impulso-resposta

- Finalmente, o dynare solta gráficos da função impulso-resposta do choque nas variáveis endógenas do modelo:



- Normalmente temos uma função para cada choque. No nosso caso tinhamos apenas um choque exógeno e quatro variáveis endógenas logo temos quatro gráficos.
  - Muito útil para entender o mecanismo de transmissão dos choques no modelo!

Outros output

- O Dynare também cria todas essas variáveis (função impulso, policy functions, etc) no workspace do matlab.
- Depois de chamar o .mod-file você pode fazer o que quiser com essas funções:
  - Comparar diferentes modelos;
  - Diferentes parâmetros;
  - Realizar sua própria simulação;
- Inclusive você pode chamar uma .mod-file do Dynare dentro de um script .m do Matlab!
- Isso pode ser útil para estimar/calibrar os parâmetros, interpretar melhor os resultados e etc.

Apêndice

Regra Geral de Loglinearização

- Com uma função multiplicativa/exponencial basta aplicar o log diretamente. Por exemplo, a função de produção é $y_t = z_t k_t^{\alpha} n_t^{1-\alpha}$:

$$\underbrace{\log(y_t) - \log(\bar{y})}_{\tilde{y}_t} = \underbrace{\log(z_t) - \log(\bar{z})}_{\tilde{z}_t} + \alpha \underbrace{\left(\log(k_t) - \log(\bar{k})\right)}_{\tilde{k}_t} + (1-\alpha) \underbrace{\left(\log(n_t) - \log(\bar{n})\right)}_{\tilde{n}_t}$$

- Com funções mais complexas isso não é possível. É necessário uma regra geral.

- Para uma regra geral, vamos utilizar uma expansão de Taylor.

- Suponha uma função, $y = f(x)$, possivelmente não-linear. Uma aproximação linear vizinhança de um ponto $\bar{x}$:

$$y = f(x) \approx f(\bar{x}) + f_x(\bar{x})(x - \bar{x})$$

Regra Geral de Loglinearização

- Imagine que você queira loglinearizar a função $z_t = f(x_t, y_t)$ na vizinhança do estado estacionário $\bar{z} = f(\bar{x}, \bar{y})$:

$$z_t = \underbrace{f(\bar{x}, \bar{y})}_{\bar{z}} + f_x(\bar{x}, \bar{y})(x_t - \bar{x}) + f_y(\bar{x}, \bar{y})(y_t - \bar{y})$$

$$\left(\frac{z_t - \bar{z}}{\bar{z}}\right) = f_x(\bar{x}, \bar{y})\frac{\bar{x}}{\bar{z}}\left(\frac{x_t - \bar{x}}{\bar{x}}\right) + f_y(\bar{x}, \bar{y})\frac{\bar{y}}{\bar{z}}\left(\frac{y_t - \bar{y}}{\bar{y}}\right)$$

$$\tilde{z}_t = \bar{x}\frac{f_x(\bar{x}, \bar{y})}{f(\bar{x}, \bar{y})}\tilde{x}_t + \bar{y}\frac{f_y(\bar{x}, \bar{y})}{f(\bar{x}, \bar{y})}\tilde{y}_t$$

- Como $\bar{x}, \bar{y}, f(\bar{x}, \bar{y})$ são funções de parâmetros, $\tilde{z}_t$ é uma função linear de $\tilde{x}_t$ e $\tilde{y}_t$.

Exemplo: Função de Produção

- Função de produção $y_t = f(z_t, k_t, n_t) = z_t k_t^{\alpha} n_t^{1-\alpha}$ no caso geral:

$$\tilde{y}_t = \overline{z}\frac{f_z(\bar{z},\bar{k},\bar{n})}{f(\bar{z},\bar{k},\bar{n})}\tilde{z}_t + \overline{k}\frac{f_k(\bar{z},\bar{k},\bar{n})}{f(\bar{z},\bar{k},\bar{n})}\tilde{k}_t + \overline{n}\frac{f_n(\bar{z},\bar{k},\bar{n})}{f(\bar{z},\bar{k},\bar{n})}\tilde{n}_t$$

- Parâmetros:

$$\begin{aligned}
\overline{z}\frac{f_z(\bar{z},\bar{k},\bar{n})}{f(\bar{z},\bar{k},\bar{n})} &= \bar{z}\frac{\bar{k}^{\alpha}\bar{n}^{1-\alpha}}{\bar{z}\bar{k}^{\alpha}\bar{n}^{1-\alpha}} = 1 \\
\overline{k}\frac{f_k(\bar{z},\bar{k},\bar{n})}{f(\bar{z},\bar{k},\bar{n})} &= \bar{k}\frac{\alpha\bar{z}\bar{k}^{\alpha-1}\bar{n}^{1-\alpha}}{\bar{z}\bar{k}^{\alpha}\bar{n}^{1-\alpha}} = \alpha \\
\overline{n}\frac{f_n(\bar{z},\bar{k},\bar{n})}{f(\bar{z},\bar{k},\bar{n})} &= \bar{n}\frac{(1-\alpha)\bar{z}\bar{k}^{\alpha}\bar{n}^{-\alpha}}{\bar{z}\bar{k}^{\alpha}\bar{n}^{1-\alpha}} = 1-\alpha
\end{aligned}$$

Volta