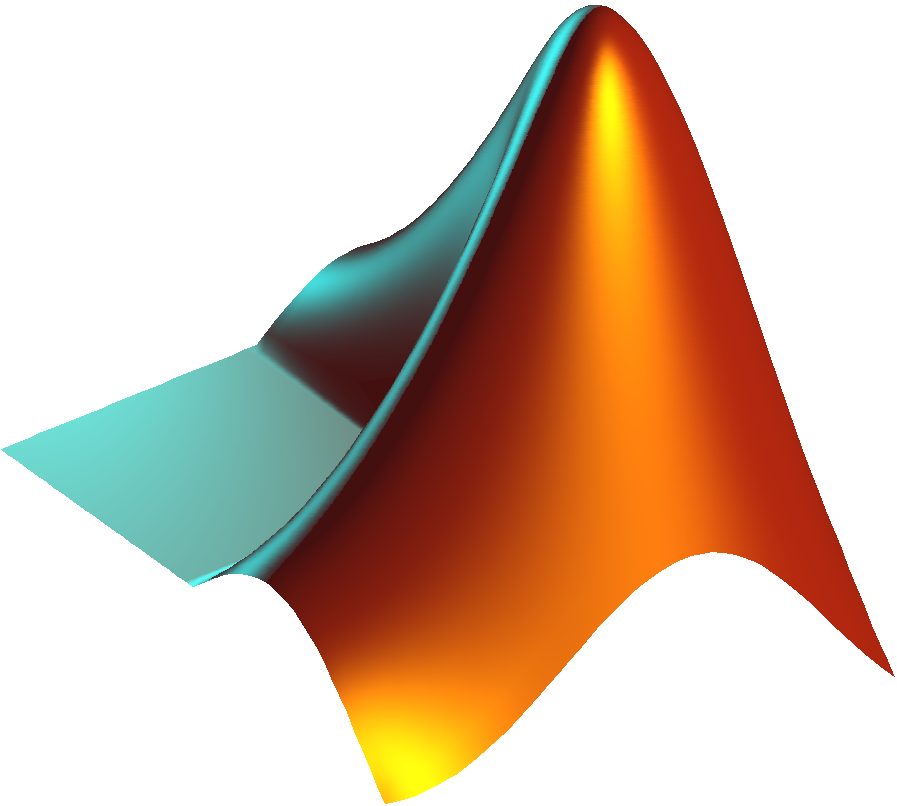
**Licenciatura em Engenharia Informática 2017/2018**

Análise e Transformação de Dados



**Análise de Série Temporal**

*dataset PL2*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Flávio Amaral Fernandes 2016240139 ffernandes@student.dei.uc.pt

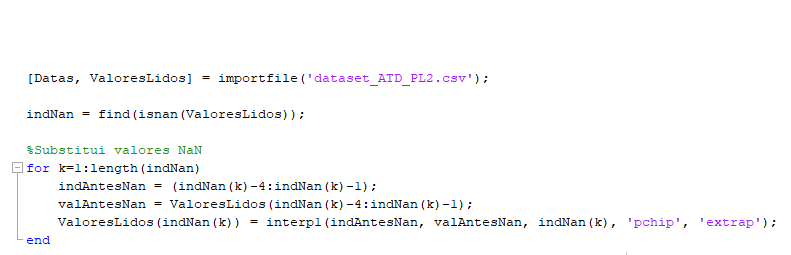
João Filipe Sequeira Montenegro 2016228672 jmontenegro@student.dei.uc.pt

# Breve Introdução

Este programa tem como finalidade analisar ao longo do tempo, os consumos diários de um conjunto de habitações, através da linguagem MATLAB.

Neste programa deverá ser possível visualizar os gráficos das tendências ao longo do tempo.

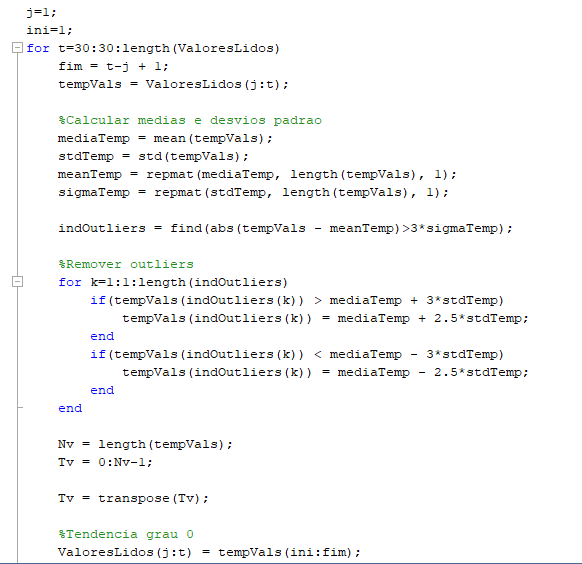
# Leitura e *NaN´s*

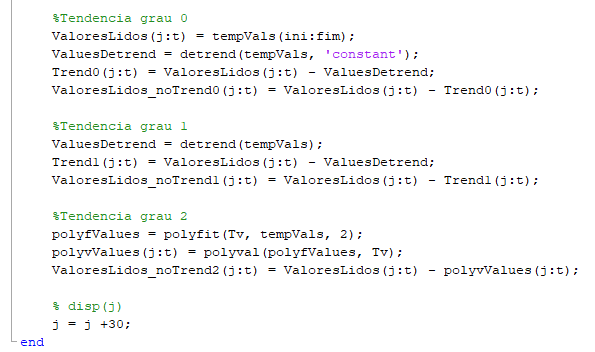


Começamos por ler os dados através do *importfile* presente no Matlab e guardamos as datas em “*Datas*” e os valores dos consumos em *“ValoresLidos”.*

Posteriormente, procuramos por valores *NaN* na matriz de *ValoresLidos* e guardamos o seu índice em “*indNaN”*. Os índices guardados serão usados para percorrer a matriz de valores e nesse índice calcular o valor mais provável usando o valor anterior e o seguinte. Este valor é estimado usando o ‘pchip’ e o ‘extrap’.

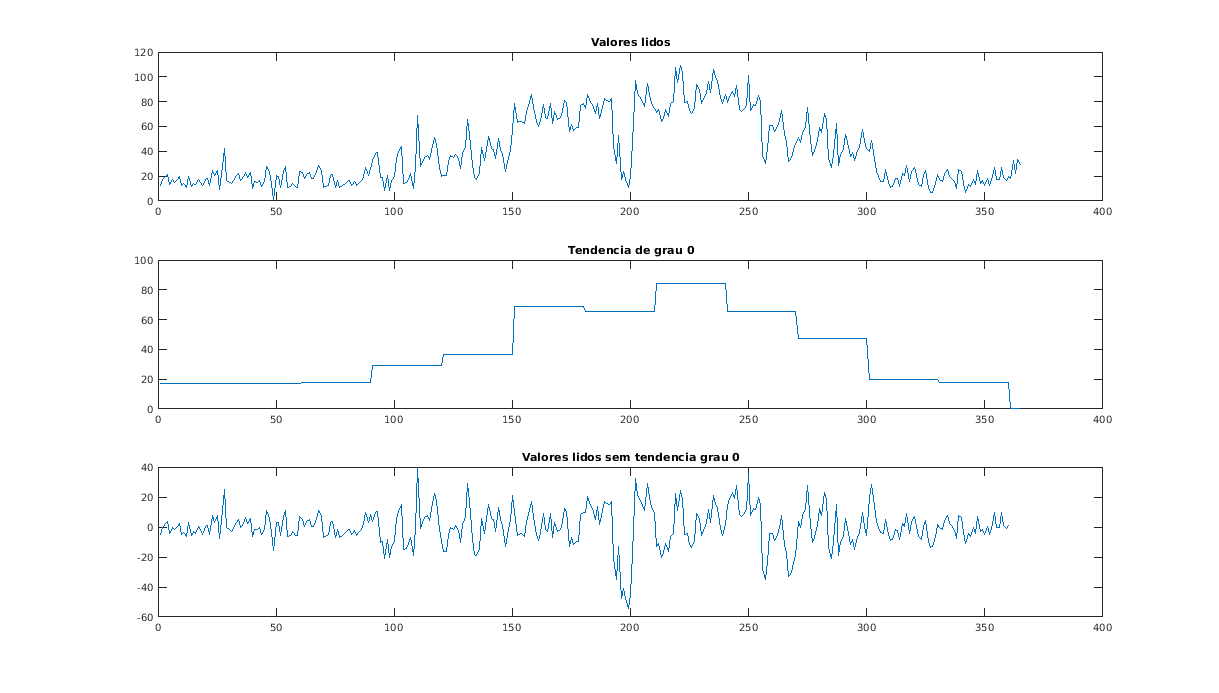
# Média, Desvio Padrão, *Outliers* e Tendências



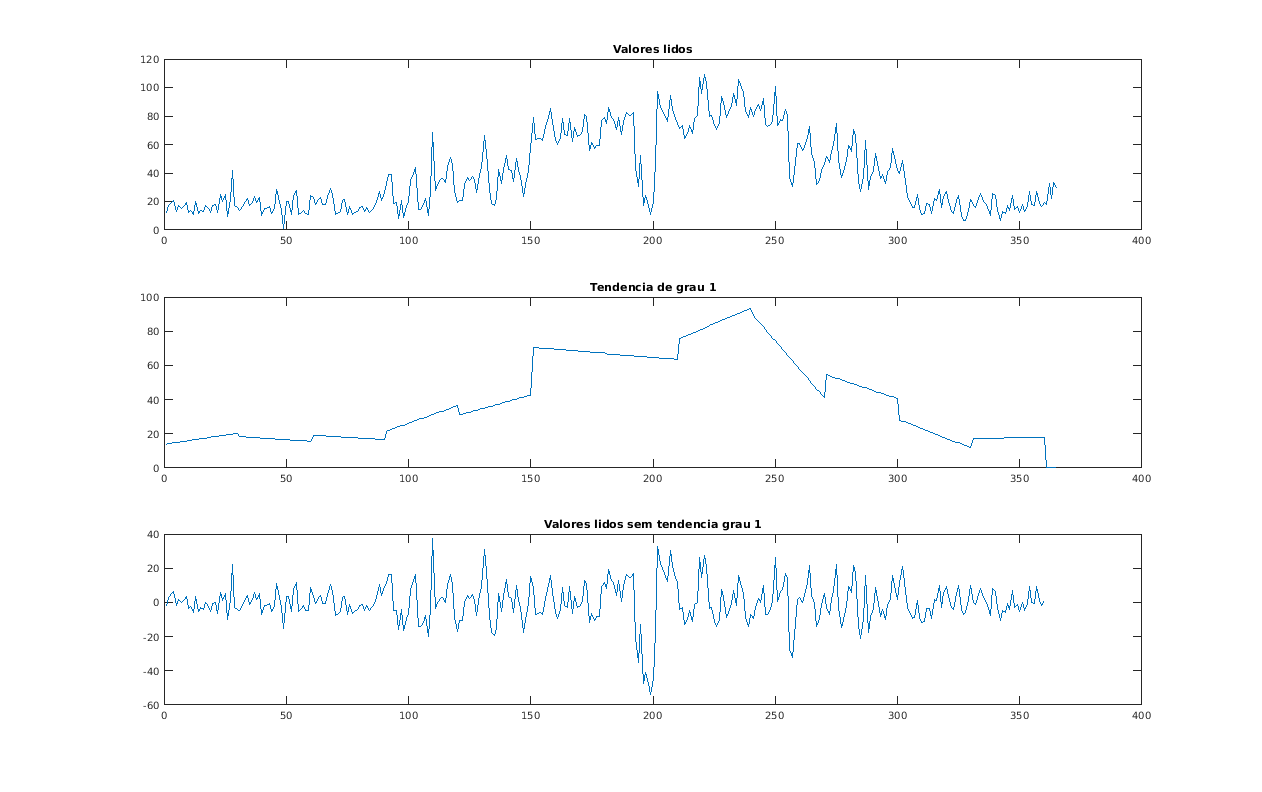


Neste ponto, calculamos todos os valores mensalmente, isto é, de 30 em 30 dias, para termos valores mais próximos da realidade. É considerado *outlier* quando o valor é superior/inferior à *mediaTemp*(média)mais/menos 3 vezes o *stdTemp*(desvio padrão).

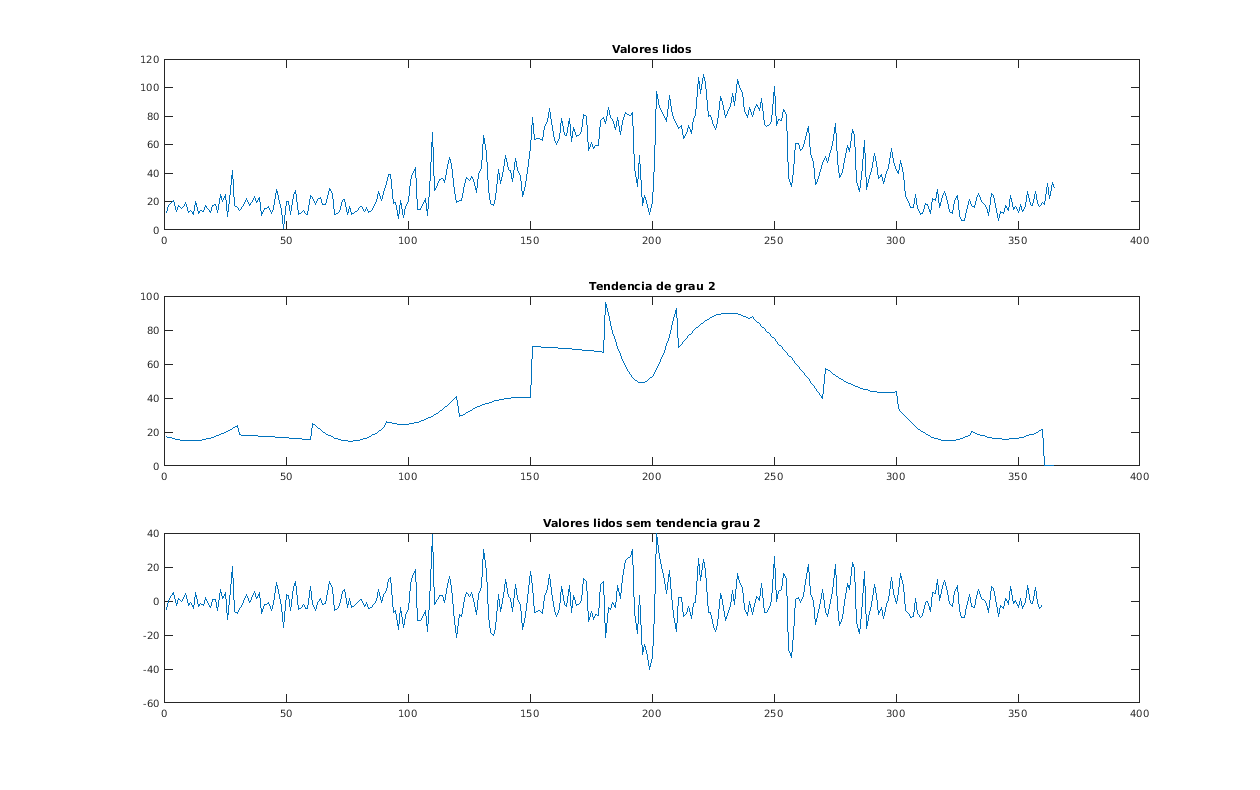
Para corrigir este problema, o valor nesse índex passa a ser a *mediaTemp* mais/menos 2,5 vezes o stdTemp. Depois disto, calculamos o 3 tendências (grau 0, 1 e 2) para verificarmos a que melhor se adequava ao nosso caso. Depois de analisar as três tendências utilizamos a tendência de grau 2.



Tendência de grau 0

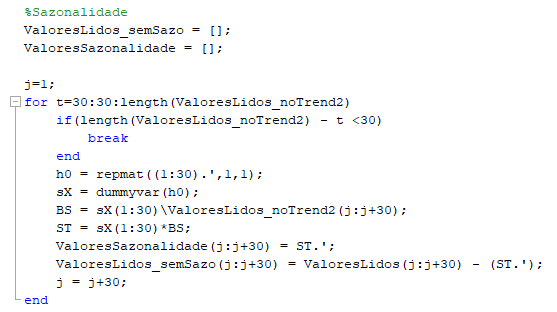


Tendência de grau 1



Tendência de grau 2

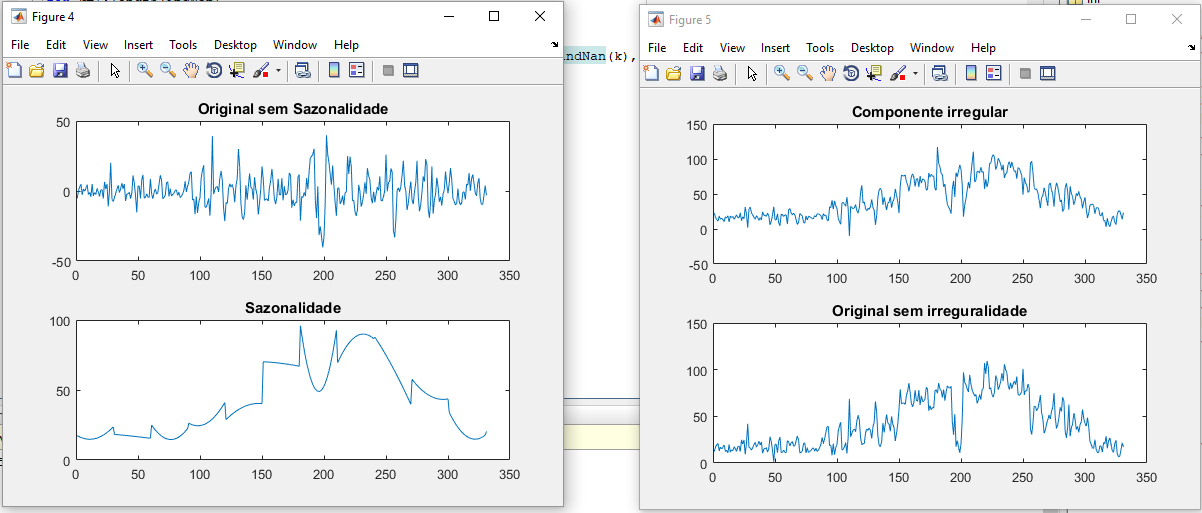
# Sazonalidade

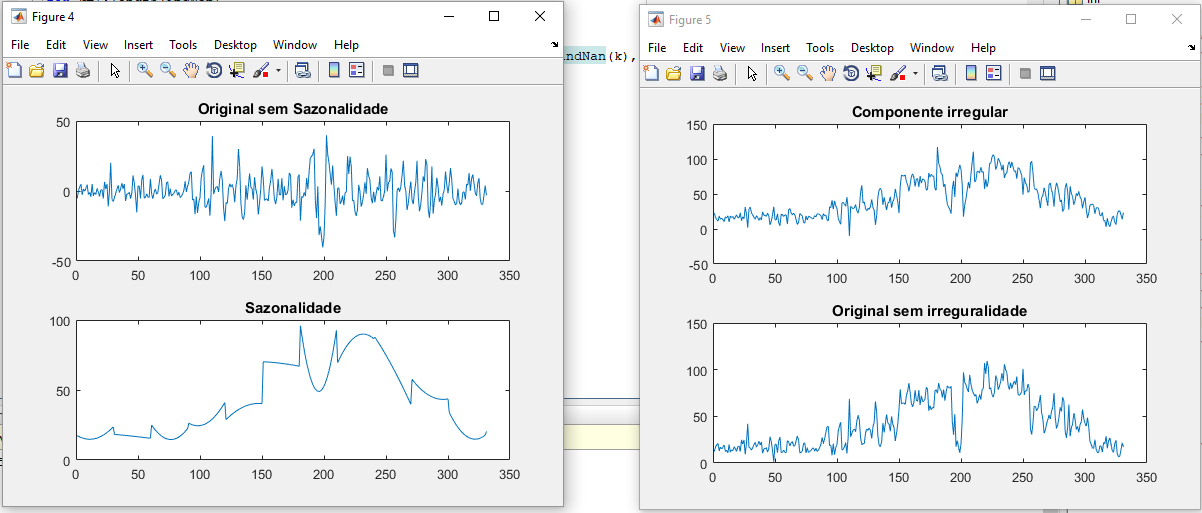


Para o cálculo da sazonalidade usamos então o valor da tendência de grau 2 por ser aquele que mais se adequa aos nossos valores. Depois voltamos a percorrer os valores da matriz a cada 30 valores para que representem os valores lidos a cada mês. Quando não existirem 30 valores para o cálculo da sazonalidade (acontece quando estamos no “último mês” da matriz) terminamos o ciclo.

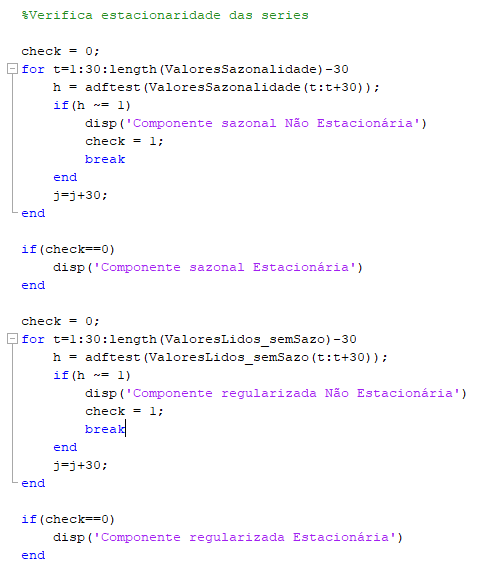
Estando a sazonalidade mensal calculada os valores são separados em duas matrizes. Uma com os valores da sazonalidade e outra com os valores lidos sem a componente sazonal.

Posteriormente representamos também a irregularidade, calculada da seguinte forma: ValoresLidos(1:331).' – ValoresSazonalidade - ValoresLidos\_noTrend2(1:331). Limitamos os valores da matriz entre 1 e 331 porque no calculo da sazonalidade dispensamos os valores quando não são 30.





# Estacionaridade



Para verificar a estacionaridade das matrizes, percorremos tanto a matriz sazonal como a regularizada e se o valor *adftest* for diferente de 1, concluímos que a componente não é estacionária e portanto, não é necessário percorrer mais e termina o ciclo.

Caso a matriz seja percorrida até ao fim e todos os valores *adftest* forem 1, podemos então concluir que a matriz é estacionária.

# Modelo AR

Para estimarmos o modelo AR da componente sazonal da série começamos por usar as funções *autocorr* e *parcorr* para obter, respetivamente, os valores da Função de Autocorrelação (FAC) e da Função de Autocorrelação Parcial (FACP), utilizando os resultados da FACP para obter o valor de *na* (explicar o valor 30?).

De seguida, é criado um objecto *iddata* com um período de amostragem correto, 1 dia, e definimos a abordagem a utilizar no modelo AR, neste caso usamos o método dos mínimos quadrados. Passamos então à estimação do modelo.

Procedemos à simulação do modelo usando a função *forecast* e os parâmetros do modelo da componente sazonal.

# Modelo ARMA

Para estimarmos agora o modelo ARMA usamos não só a variável obtida por FACP (*na*), mas também *nc*, obtida por FAC.

De seguida, usando as funções *armax* e *armaxOptions* definimos os parâmetros do modelo, o método de procura será a opção automática.

# Modelo ARIMA