QF627 Programming and Computational Finance

HWS0102: MATLAB Basics

Q1: HDB Loan Calculator GUI, which defines and uses function **funP** to compute the monthly installment. Remove all the comments in the auto-generated code.

|  |  |
| --- | --- |
| funP.m | %234567890123456789012345678901234567890123456789012345678901234567890  function [P]=funP(PV, r, t)  P=(r/12\*PV)/(1-(1+r/12)^(-12\*t));  end |
| Pushbutton Callback in HDBLoanCalculator.m | function varargout = untitled(varargin)  gui\_Singleton = 1;  gui\_State = struct('gui\_Name', mfilename, ...  'gui\_Singleton', gui\_Singleton, ...  'gui\_OpeningFcn', @untitled\_OpeningFcn, ...  'gui\_OutputFcn', @untitled\_OutputFcn, ...  'gui\_LayoutFcn', [] , ...  'gui\_Callback', []);  if nargin && ischar(varargin{1})  gui\_State.gui\_Callback = str2func(varargin{1});  end  if nargout  [varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});  else  gui\_mainfcn(gui\_State, varargin{:});  end  function untitled\_OpeningFcn(hObject, eventdata, handles, varargin)  handles.output = hObject;  guidata(hObject, handles);  function varargout = untitled\_OutputFcn(hObject, eventdata, handles)  varargout{1} = handles.output;  function edit\_LoanAmount\_Callback(hObject, eventdata, handles)  function edit\_LoanAmount\_CreateFcn(hObject, eventdata, handles) |

Q2: Write a MATLAB function for the computation of Income Tax (ver. 2), with the following inputs/outputs

Inputs: x, bi, mi and xi

Output: tax

|  |  |
| --- | --- |
| Python | **bi=[0, 200, 550, 3350, 7950, 13950, 21150, 28750, 36550, 44550]**  **mi=[2.0, 3.5, 7.0, 11.5, 15.0, 18.0, 19.0, 19.5, 20.0, 22.0]**  **xi=[20000, 30000, 40000, 80000, 120000, 160000, 200000, 240000,**  **280000, 320000]**  **x=400\_000**  **if x>xi[-1]:**  **i=len(xi)-1**  **else:**  **i=next(filter(lambda w: w[1]>x, enumerate(xi)))[0]-1**    **if i==-1:**  **y=0**  **else:**  **y=bi[i]+mi[i]/100\*(x-xi[i])**  **print(y)** |
| MATLAB function | %2345678901234567890123456789012345678901234567890123456789012345 |
| MATLAB  script |  |

Q3: Write **a MATLAB function** for solving the Pandigital Formula problem, with the following inputs/outputs

Inputs: year

Outputs: None. (The function print all formula and “Done!”.)

|  |
| --- |
| %234567890123456789012345678901234567890123456789012345678901234567890123456789  year = 2017  o={'+','-','\*','/',''}  for o1 = o  for o2 = o  for o3 = o  for o4 = o  for o5 = o  for o6 = o  for o7 = o  for o8 = o  s=['1', o1{1}, ...  '2', o2{1}, ...  '3', o3{1}, ...  '4', o4{1}, ... |

Q4: Write **a MATLAB function** for the Sudoku Solving, with the following inputs/outputs

Inputs: the puzzle as an 81-digit character vector

Outputs: solution of the puzzle

|  |  |
| --- | --- |
| Python | def same\_row(i,j): return (i//9)==(j//9)  def same\_col(i,j): return (i%9)==(j%9)  def same\_block(i,j): return ((i//27)==(j//27)) and (((i%9)//3)==((j%9)//3))  def r(s):  i=s.find('0')  if i==-1:  print(s)  else:  excluded\_numbers={s[j] for j in range(81) if same\_row(i,j)  or same\_col(i,j)  or same\_block(i,j)}  for m in set('123456789')-excluded\_numbers:  r(s[:i]+m+s[i+1:])  s=('390060807' + '020030050' + '000005096' + '900502400' + '000000000'  '003907002' + '810600000' + '030050080' + '502090043')  print(s)  print(r(s)) |
| MATLAB M-File Function | %23456789012345678901234567890123456789012345678901234567890123456789012345 |
| M-File  Script |  |

Q5: Complete the European Call Option Class and run the same example as the Python’s.

|  |  |
| --- | --- |
| Python | from math import log, sqrt, exp  from scipy import stats  class call\_option(object):  def \_\_init\_\_(self, S0, K, T, r, sigma):  self.S0 = float(S0)  self.K = K  self.T = T  self.r = r  self.sigma = sigma  def value(self):  d1 = ((log(self.S0 / self.K) + (self.r + 0.5 \* self.sigma \*\* 2) \* self.T)  / (self.sigma \* sqrt(self.T)))  d2 = ((log(self.S0 / self.K) + (self.r - 0.5 \* self.sigma \*\* 2) \* self.T)  / (self.sigma \* sqrt(self.T)))  value = (self.S0 \* stats.norm.cdf(d1, 0.0, 1.0)  - self.K \* exp(-self.r \* self.T) \* stats.norm.cdf(d2, 0.0, 1.0))  return value  def vega(self):  d1 = ((log(self.S0 / self.K) + (self.r + 0.5 \* self.sigma \*\* 2) \* self.T)  / (self.sigma \* sqrt(self.T)))  vega = self.S0 \* stats.norm.cdf(d1, 0.0, 1.0) \* sqrt(self.T)  return vega  def imp\_vol(self, C0, sigma\_est=0.2, it=100):  option = call\_option(self.S0, self.K, self.T, self.r, sigma\_est)  for i in range(it):  option.sigma -= (option.value() - C0) / option.vega()  return option.sigma  o=call\_option(100., 105., 1.0, 0.05, 0.2)  print(o.value())  print(o.vega())  print(o.imp\_vol(C0=value)) |
| MATLAB M-File for the class | | %234567890123456789012345678901234567890123456789012345678901234567890123456789012 |
| MATLAB M-File for the class (continued) |  |
| M-File Script |  |