Latex Template

Hunter Kruger-Ilingworth (14198489)

November 30, 2023

Contents

List of Figures

List of Tables

1 Code

1.1 Python

Listing 1: Python code to create an incidence matrix

```
import numpy as np
def incmatrix(genl1,genl2):
    m = len(genl1)
    n = len(genl2)
    M = None #to become the incidence matrix
    VT = np.zeros((n*m,1), int)  #dummy variable
    x = 0
    #compute the bitwise xor matrix
    M1 = bitxormatrix(genl1)
    M2 = np.triu(bitxormatrix(genl2),1)

    for i in range(m-1):
        for j in range(i+1, m):
            [r,c] = np.where(M2 == M1[i,j])
            for k in range(len(r)):
                VT[(i)*n + r[k]] = 1;
                VT[(i)*n + c[k]] = 1;
                VT[(j)*n + r[k]] = 1;
                VT[(j)*n + c[k]] = 1;

                if M is None:
                    M = np.copy(VT)
                else:
                    M = np.concatenate((M, VT), 1)

                VT = np.zeros((n*m,1), int)

    return M
```

1.2 Matlab

Listing 2: MATLAB Code that tests Solar System Function

```
function test_advanced_level(unit_under_test)
    % TEST_ADVANCED_LEVEL Test the simulator against the advanced level of
    achievement.
    %
    % TEST_ADVANCED_LEVEL(@unit) tests a function called "unit" instead of the
    % default, "solarsystem".
    %
    % This is provided as a means for your to test your program's accuracy. We
    % supply here high precision answers that can use as a benchmark against
    % which to compare your code.
    %
    % A program similar to this one will be used during marking to test your
    % program's accuracy and speed. We'll use different initial conditions, so
    % don't try simply hard-coding these answers! :)
    %

    % Default to a function named "solarsystem"
    if nargin < 1
        unit_under_test = @solarsystem;
    end

    % Data
    % Data source: NASA JPL Development Emphemeris DE405, imported into Matlab
    % using https://au.mathworks.com/matlabcentral/fileexchange/46074-jpl-
    ephemeris-manager
    mass = [1.98879724324801e+30;3.30167548185139e+23;4.86825414184162e
    +24;5.97333182929537e+24;6.41814989746695e+23;1.89888757501372e
    +27;5.68569250232054e+26;8.68357411676561e+25;1.02450682828011e
    +26;1.47100387814202e+22];
    p = [-410978934.937975 -52564098.573049
    -11647539.5911275;-20263704896.5463 37298969437.5484
    21998926177.1807;107457059203.846 12751258164.7855
    -1081247256.91775;-104473131433.549 95807463843.1787
    41554965796.5625;-47532402438.2755 -197479402904.819
    -89286739068.5338;740812325977.265 -29623952257.2314
    -30753799138.017;-391719672964.493 1189107854643.27
    507856891148.711;-2396814857836.84 -1270773906334.37
    -522608874439.045;-1545201887440.28 -3957617757444.78
    -1581427940931.15;-4371341308972.33 -1084064015240.84 978703610774.062];
    v = [1.94673233456669 -10.8814016462929 -4.7775329435922;-54017.2779417951
     -18415.0969798133 -4228.50548119061;-3793.57777814318 31524.0648690534
    14419.9306824639;-21597.9402281813 -19392.9951239518
    -8410.50277824797;24596.1594690375 -2563.11636886769
    -1841.7251251432;538.777252737696 12558.0983493514
    5370.16231719295;-9767.15104601119 -2764.87492216388
    -721.832483731844;3335.76872430951 -5686.29309895411
    -2537.72389267233;5074.99185394443 -1640.69964089467
    -797.853610190395;1586.81468930053 -5301.34210829372 -2132.29213550457];

    % Use inner planets only; supply them in the order Sun, Earth, Mercury,
    % Venus, Mars (so that colours used for the Sun and Earth in the other
```

```
30      % tests apply here too)
31      i = [1 4 2 3 5];
32      mass = mass(i);
33      p = p(i,:);
34      v = v(i,:);
35
36      % Test 1
37      Test_3D_Solar_System(false);
38
39      % Test 2
40      Test_3D_Solar_System(true);
41
42          function test_result(parameter, value, units, comparator, benchmark)
43              if strcmp(units, '%')
44                  fprintf('  %28s :  %-15.6f', [parameter ' (' units ')'], value
    );
45              else
46                  fprintf('  %28s :  %-15.6g', [parameter ' (' units ')'], value
    );
47              end
48
49              if nargin == 5
50                  if comparator(value, benchmark)
51                      fprintf('   ** PASS. Meets or exceeds the expectation of %
    g%s', benchmark, units);
52                  else
53                      fprintf('   ** FAIL. Does not meet the expectation of %g%s
    ', benchmark, units);
54                  end
55              end
56              fprintf('\n');
57          end
58
59
60          function Test_3D_Solar_System(speed_test)
61              if speed_test
62                  fprintf('<strong>*** [Advanced level] Inner planets in 3D (
    execution speed test)</strong>\n');
63              else
64                  fprintf('<strong>*** [Advanced level] Inner planets in 3D</
    strong>\n');
65              end
66
67              % Run the program
68              tic();
69              [final_p, final_v] = unit_under_test(p, v, mass, 400*24*60*60,
    speed_test);
70              t = toc();
71              test_result('Execution time', t, 's');
72
73              % Check the dimensions of the return values
74              assert(all(size(final_p) == size(p)), 'Expected size of return
    value "p" to be %ix%i, received %ix%i instead.', size(p,1), size(p,2),
    size(final_p, 1), size(final_p, 2));
```

```
75          assert(all(size(final_v) == size(v)), 'Expected size of return
    value "v" to be %ix%i, received %ix%i instead.', size(v,1), size(v,2),
    size(final_p, 1), size(final_p, 2));
76
77          % Check the answers
78          correct_p = [-345966512.946938 -427734515.389726
    -176305011.39972;-146624134718.92 23657267681.4718
    10263838966.7501;18103318019.8 -57037628697.2036
    -32327489902.5598;32774480992.525 -94215943100.9452
    -44459425421.2662;-136512481872.172 183605819453.111 87921613716.9978];
79          correct_v = [1.80779317569275 -10.8097599943337
    -4.73922210995349;-5754.8524496641 -27006.432527836
    -11711.495344779;37086.6176187111 15367.1705312566
    4360.22910247245;33125.9176644671 10361.9441289187
    2564.05625302394;-19228.3713286498 -10562.8647670158 -4323.54314113474];
80
81          % mercury is harder to simulate because it moves the fastest
82          % the objects are in this order: Sun, Earth, Mercury, Venus, Mars
83          expectations = [0.1 1 5 1 1];
84          for i = 1:size(p,1)
85              test_result(sprintf('Object %i position error', i), norm(
    final_p(i,:) - correct_p(i,:))/norm(correct_p(i,:))*100, '%', @le,
    expectations(i));
86              test_result(sprintf('Object %i velocity error', i), norm(
    final_v(i,:) - correct_v(i,:))/norm(correct_v(i,:))*100, '%', @le,
    expectations(i));
87          end
88      end
89
90  end
```

It can clearly be seen from listings 1 to 2 that code chunks are cool and sick.

2 Figure Rendering

2.1 3D Plots

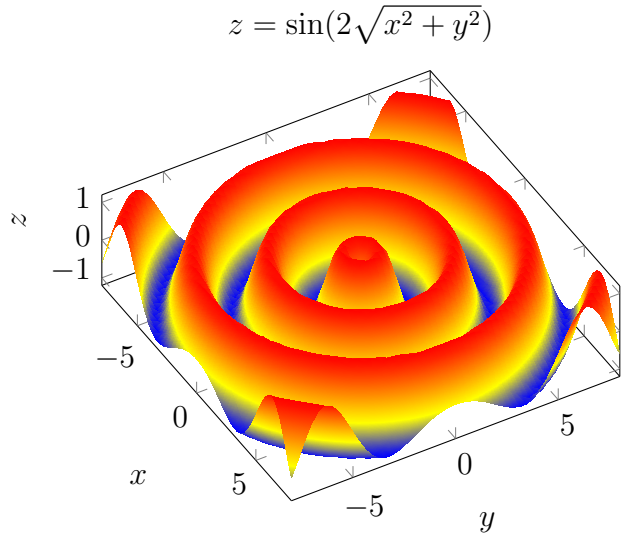

Fig. 1. 3D plot of $z = \sin(2\sqrt{x^2 + y^2})$

3 Table input from csv

Country	Population	Area	Oil Prod.	GDP	Education	Roadways	Net Users
Australia	23232413	7741220	289700	50300	5.6	0.106342024	60%
Canada	35623680	9984670	3679000	48300	5.4	0.10439003	60%
France	67106161	643801	16420	43800	5.9	1.597459463	60%
Germany	80594017	357022	46590	50400	5.1	1.806611357	60%
Italy	62137802	301340	70670	38100	4.5	1.618437645	60%
Japan	126451398	377915	3918	42800	3.8	3.224989746	60%
Russia	142257519	17098242	10550000	27800	4.1	0.075059588	60%
United Kingdom	65648100	243610	933000	44100	6.2	1.619096096	60%
United States	326625791	9833517	8853000	59500	5.4	0.669812235	60%

Table 1: Your Table Caption