Exam information	
Course code and name	COSC2500 Numerical Methods in Computational Science
Semester	Semester 2, 2020
Exam type	Online, non-invigilated
Exam date and time	Please refer to your personalised timetable
Exam duration	You have a 12-hour window in which you must complete your exam. You can access and submit your exam at any time within the 12 hours. Even though you have the entire 12 hours to complete and submit your exam, the expectation is that it will take most students between 2 and 2.5 hours.
Reading time	Reading time has not been formally allocated for online exams; however, students are encouraged to review and plan their approach for the exam before they start. The total exam time should be sufficient to do this.
Exam window	You must commence your exam during the time listed in your personalised timetable. The exam will remain open **only** for the duration of the exam.
Weighting	This exam is weighted at 30% of your total mark for this course.
Permitted materials	This is an open book exam – all materials permitted You may use any of the learning resources provided via the COSC2500 Blackboard site; you may access these via Blackboard during the exam. You may use your own notes and project reports, software you have used in the course, the textbook, any calculator, and a bilingual dictionary.
Instructions	Five questions, correctly answered, from any part of the exam, are required for a grade of 5 for this exam (four questions for a grade of 4). For a grade of 6 or 7 (1-12 additional marks), you must answer additional questions, in addition to the five required for a grade of 5, from Part 2. Up to 6 marks can be obtained from each additional question. You can print the exam and write your answers on blank paper, or write electronically on a suitable device, and copy-and-paste graphs and other output from your software. Scan or photograph your work if necessary and upload your answers to Blackboard as a single pdf file. You must submit your answers as a single electronic file through Blackboard before the end of the allowed time. You should include your **name and student number** on the first page of the file that you submit
Who to contact	If you have any concerns or queries about a particular question, or need to make any assumptions to answer the question, state these at the start of your solution to that question. You may also include queries you may have made with respect to a particular question, should you have been able to 'raise your hand' in an examination room. If you experience any **technical difficulties** during the exam, contact the Library AskUs (https://web.library.uq.edu.au/contact-us) service for advice (open 7am–10pm, 7 days a week, Brisbane time): Chat: https://support.my.uq.edu.au/app/chat/chat_launch_lib/p/45 Phone: +61 7 3506 2615 Email: examsupport@library.uq.edu.au You should also ask for an email documenting the advice provided so you can provide this on request. In the event of a **late submission**, you will be required to submit evidence that you completed the exam in the time allowed. We recommend you use

	a phone camera to take photos (or a video) of every page of your exam. Ensure that the photos are time-stamped. If you submit your exam after the due time then you should send details (including any evidence) to SMP Exams (exams.smp@uq.edu.au) as soon as possible after the end of the exam.
Important exam condition information	The normal academic integrity rules apply. You cannot cut-and-paste material other than your own work as answers. You are not permitted to consult any other person – whether directly, online, or through any other means – about any aspect of this assessment during the period that this assessment is available. If it is found that you have given or sought outside assistance with this assessment then that will be deemed to be cheating and will result in disciplinary action. By undertaking this online assessment, you will be deemed to have acknowledged UQ's academic integrity pledge by making the following declaration: *"I certify that my submitted answers are entirely my own work and that I have neither given nor received any unauthorised assistance on this assessment item"*.

Part 1—Required

The primary purpose of Part 1 is for students to demonstrate sufficient knowledge and understanding to earn a grade of 4 or 5. Part 1 questions will not *earn marks towards grades of 6 or 7.*

1. Numerically calculate the derivative of $f(x) = x^5 - 4x + 3$ at $x = 1$, for step sizes of $h = 1 \times 10^{-3}$, $h = 1 \times 10^{-5}$, and $h = 1 \times 10^{-7}$. Are the errors what you expected them to be? (The exact value is $f'(x = 1) = 1$.) Include your code and output in your answer.

2. Numerically find the maximum of $f(x) = x^3 \exp(-x^2)$. At what value of x is the maximum? What is the value of $f(x)$ at the maximum? Include your code and output in your answer.

3. Solve $H_8 x = b$ where H_8 is the 8×8 Hilbert matrix, with $H_{ij} = 1/(i + j - 1)$, and b is a vector of all ones. You can use the Matlab function `hilb()` to generate the Hilbert matrix. Estimate the error. Include your code and output in your answer.

4. Since a matrix inversion operation is $O(N^3)$, i.e., order N^3, where the matrix is $N \times N$ in size, it becomes impractical to solve large linear systems by matrix inversion. Discuss what other methods can be used and their advantages and disadvantages.

5. Numerically solve the differential equation $f'(x) = x \sin(x)$ from $x = 0$ to $x = \pi$, with the boundary condition $f(0) = 0$. Include your code and output in your answer.

6. Why can stiff systems cause difficulty when solving systems of ODEs using adaptive step-size methods? What can be done to deal with these difficulties?

7. Why is it usually better to perform a least-squares fit to a lower-order polynomial than an exact fit to a high-order polynomial?

8. Numerically integrate $f(x, y, z) = (x + 2) \sin(\pi y/2)\sqrt{z}/3$ for $x = 0$ to 1, $y = 0$ to 1, and $z = 0$ to 1. The minimum and maximum values of $f(x, y, z)$ in this interval are 0 and 1, so you can modify your Monte Carlo integration code from the Module 6 assignment to calculate this integral. Estimate your error. Include your code and output in your answer.

Part 2—Additional

The primary purpose of Part 2 is for students to demonstrate sufficient knowledge and understanding to earn a grade of 6 or 7, after satisfying the requirements for a grade of 5. Additional questions from Part 2 can earn marks towards grades of 6 or 7 if answers demonstrate achievement appropriate for grades of 6 or 7.

- *Additional questions on Part 2 are worth 6 marks each.*
- *A maximum of 12 marks can be obtained.*

9. Discuss how you could computationally model the baking of a cake, incorporating the changes in density, moisture, thermal conductivity and heat capacity during baking.

10. Discuss a specific example of validation and verification of a model and its computational solution.

11. How can computational error be estimated without comparing results with known analytical results, experimental measurements, or other computational results? That is, how can we estimate the error in a computational result using only the program that produced that result? Give a computational example.

12. Sometimes we solve differential equations numerically, even when a general solution of the differential equation is known. Not only do we need to satisfy the differential equation, but we also need to satisfy the boundary conditions. How can we use a known general solution in our numerical solution?

13. Discuss, with critical analysis, a paper from the research literature.

14. Discuss one of the topics presented in the COSC2500 guest lecture series on numerical methods in research.

15. A computational linguist at The University of Queensland has been writing C code to distinguish between different European languages. Frustratingly, her code still has four problems—one will stop the program from compiling, one is a bug in the "score" function, and there are two other errors. Describe and fix the problems in her code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct text_prob { char bigram[2]; int log; };
typedef struct text_prob prob;

prob en[] = // Language statistics for English, German and French
{{"TH",348},{"HE",336},{"ER",289},{"IN",289},{"AN",281},
{"RE",265},{"ES",256},{"ND",251},{"EN",246},{"ON",245},
{"ED",245},{"ST",243},{"NT",240},{"TO",238},{"AT",237},
{"HA",233},{"EA",233},{"OU",223},{"OR",223},{"OF",221}};
prob de[] =
{{"EN",368},{"ER",357},{"CH",337},{"DE",309},{"EI",306},
{"ND",305},{"TE",296},{"IN",289},{"IE",283},{"GE",276},
{"ES",268},{"UN",263},{"NE",257},{"HE",247},{"IC",246},
{"AN",240},{"BE",239},{"ST",238},{"RE",237},{"SE",235}};
prob fr[] =
{{"ES",339},{"LE",309},{"RE",307},{"EN",306},{"DE",300},
{"NT",276},{"ER",275},{"TE",274},{"ET",273},{"ON",270},
{"AI",269},{"OU",267},{"SE",264},{"EL",259},{"AN",249},
{"LA",249},{"IT",249},{"QU",247},{"NE",243},{"ME",235}};

char *text[] = {
"THEQUICKBROWNFOXJUMPSOVERTHELAZYDOG",
"CEQUINESTPASCLAIRNESTPASFRANCAISCEQUINESTPASCLAIREST"
"ENCOREANGLAISITALIENGRECOULATIN",
"EINECHTERDEUTSCHERMANNMAGKEINENFRANZENLEIDEN"
"DOCHIHREWEINETRINKTERGERN",
};

int score (char *string, prob input[])
{
  int sum = 0, ii, *map = calloc(26*26, sizeof(int));
  for (ii = 0; ii < 20; ii++)
    map[26*(input[ii].bigram[0] - 'A') +
                  input[ii].bigram[1] - 'A'] = input[ii].log;

  for (ii = 0; ii < strlen(string) - 1; ii++)
    sum += map[26*string[ii] - 'A' + string[ii + 1] - 'A'];
  return sum;
}

int
main ()
```

```
{
  char *lang[] = { "English", "German", "French" };
  for (int ii = 0; ii < 3; ii++)
    {
      int best = score(text[ii], en);
      int sc = score(text[ii], de);
      if (sc > best) { int which = 1; best = sc; }
      sc = score(text[ii], fr);
      if (sc > best) which = 2;
      printf("%s is %s\n", text[ii], lang[which]);
    }
}
```

16. Discuss and critique the following Matlab code:

```
stop_time = 100000;

% Spherical coordinates with spherical symmetry gives
%       1D, r in [0,inf)
%
% Variables:
% op_force(r) = optical force, turned on at t=0,
%                 turned off at t = time_off
% peo_num(r) = number of peo molecules in finite volume element
%                with outer boundary at r
% volume(r) = volume of finite volume element with
%                  outer boundary at r
% peo_conc(r) = peo concentration, peo_conc = peo_num ./ volume
% radius(r) = actual radius, with r being an index

% Optical force constant
ofc = 10;

% Pressure force constant
pfc = 1;

radius = linspace(0,10,100);
radius = radius(2:end);

% CHANGE THIS TO A VECTOR IF NON-UNIFORM SPACING IS USED
dr = radius(2) - radius(1);

volume = 4*pi/3 * radius.^3;
volume = [ volume(1) diff(volume) ];

% Set initial concentration to 1
peo_num = volume;
peo_conc = ones(size(peo_num));

% Optical force
w = 2;
op_force = - 2 * radius/w^2 .* exp( - radius.^2/w^2 );

t = 0;
```

```
40
41 dn = zeros(size(peo_num));
42
43 tv = 0;
44 cmax = 1;
45 cmin = 1;
46
47 n = 0;
48
49 n = 0;
50
51 % Set times to save a snapshot. Turn off by setting save_n = 0
52 save_times = [ 3.3 21.4 171.6793 1187 8400 ];
53 save_n = 1;
54 save_max = length(save_times);
55
56 while t < stop_time
57
58     % Concentration at edge?
59     edge_conc =  ( peo_conc + [ peo_conc(2:end) 1 ] )/2;
60
61     % Pressure force?
62     p_force = ( peo_conc - [ peo_conc(2:end) 1 ] ) ./ dr;
63
64     % Rate of change of number in volume
65     dn(1) = - ( ofc * op_force(1) + pfc * p_force(1) ) ...
66         * edge_conc(1);
67     dn(2:end) = ( ofc * op_force(1:(end-1)) + pfc ...
68         * p_force(1:(end-1)) ) .* edge_conc(1:(end-1)) ...
69         - ( ofc * op_force(2:end) + pfc * p_force(2:end) ) ...
70         .* edge_conc(2:end);
71
72     % Choose time step to allow X% change in number
73     % in any volume element
74     dt = min(abs( 0.004 * peo_num ./ dn ));
75     dt = min(dt,1);
76
77     peo_num = peo_num + dt * dn;
78     peo_conc = peo_num ./ volume;
79
80     t = t + dt;
81
82     n = n + 1;
83
84     if n/1000 == round(n/1000)
85
86         tv = [ tv t ];
87         cmax = [ cmax max(peo_conc) ];
88         cmin = [ cmin min(peo_conc) ];
89
90     end
91
92     if save_n > 0 & save_n < save_max + 1
```

```
        if t > save_times(save_n)
            snapshots{save_n} = peo_conc;
            times(save_n) = t;
            fprintf(1,'Snapshot saved: t = %f, n = %d\n',t,n);
            save_n = save_n + 1;
        end
    end

end

% If saving, add on a snapshot of the final state
if save_n > 0
    snapshots{save_n} = peo_conc;
    times(save_n) = t;
end

% save peo.mat
```

17. Discuss and critique the following C code:

```
#include <stdio.h>
#include <math.h>

#define N 5
double Pi;
double lroots[N];
double weight[N];
double lcoef[N + 1][N + 1] = {{0}};

void lege_coef()
{
    int n, i;
    lcoef[0][0] = lcoef[1][1] = 1;
    for (n = 2; n <= N; n++) {
        lcoef[n][0] = -(n - 1) * lcoef[n - 2][0] / n;
        for (i = 1; i <= n; i++)
            lcoef[n][i] = ((2 * n - 1) * lcoef[n - 1][i - 1]
                    - (n - 1) * lcoef[n - 2][i] ) / n;
    }
}

double lege_eval(int n, double x)
{
    int i;
    double s = lcoef[n][n];
    for (i = n; i; i--)
        s = s * x + lcoef[n][i - 1];
    return s;
}

double lege_diff(int n, double x)
{
    return n * (x*lege_eval(n,x) - lege_eval(n-1,x))/(x*x - 1);
}
```

```
35
36 void lege_roots()
37 {
38     int i;
39     double x, x1;
40     for (i = 1; i <= N; i++) {
41         x = cos(Pi * (i - .25) / (N + .5));
42         do {
43             x1 = x;
44             x -= lege_eval(N, x) / lege_diff(N, x);
45         } while ( fdim( x, x1) > 2e-16 );
46         /*  fdim( ) was introduced in C99, if it isn't available
47          *  on your system, try fabs( ) */
48         lroots[i - 1] = x;
49
50         x1 = lege_diff(N, x);
51         weight[i - 1] = 2 / ((1 - x * x) * x1 * x1);
52     }
53 }
54
55 double lege_inte(double (*f)(double), double a, double b)
56 {
57     double c1 = (b - a) / 2, c2 = (b + a) / 2, sum = 0;
58     int i;
59     for (i = 0; i < N; i++)
60         sum += weight[i] * f(c1 * lroots[i] + c2);
61     return c1 * sum;
62 }
63
64 int main()
65 {
66     int i;
67     Pi = atan2(1, 1) * 4;
68
69     lege_coef();
70     lege_roots();
71
72     printf("Roots: ");
73     for (i = 0; i < N; i++)
74         printf(" %g", lroots[i]);
75
76     printf("\nWeight:");
77     for (i = 0; i < N; i++)
78         printf(" %g", weight[i]);
79
80     printf("\nintegrating Exp(x) over [-3, 3]:\n\t%10.8f,\n"
81         "compred to actual\n\t%10.8f\n",
82         lege_inte(exp, -3, 3), exp(3) - exp(-3));
83     return 0;
84 }
```

Formula sheet

Taylor series

$$f(x+h) = f(x) + \frac{f'(x)h}{1!} + \frac{f''(x)h^2}{2!} + \frac{f'''(x)h^3}{3!} + \frac{f''''(x)h^4}{4!} + ... \tag{1}$$