This question paper consists of 9 printed pages, each of which is identified by the Code Number COMP5930M.

This is a closed book examination. No material is permitted.

School of Computing

January 2019

COMP5930M

Scientific Computation

Answer ALL questions

Time allowed: 2 hours

NOTE TO INVIGILATOR AND STUDENT

This question paper must be attached with a treasury tag to the back of the answer booklets. It is the student's responsibility to attach the question paper.

Question 1

Assume that $F(x) : \mathbb{R} \rightarrow \mathbb{R}$ is a continuously differentiable scalar function that has the derivative $F'(x)$. We want to find a solution x^* to the nonlinear equation $F(x^*) = 0$.

To find the solution, we consider two different numerical methods, *Newton's method* or the *bisection method*. The pseudocode for these two methods is provided below. The methods require either a starting point x_0 or an initial bracket $[x_L, x_R]$, a convergence tolerance Tol, and the maximum number of iterations $k_{\max}$.

Newton's method

[x,f] = Newton(F, x_0, Tol, $k_{\max}$)

A1. Set $k = 0$

A2. While $|F(x_k)| > Tol$ and $k < k_{\max}$

- (i). Calculate $F'(x_k)$
- (ii). Update $x_{k+1} = x_k - F(x_k)/F'(x_k)$
- (iii). Increment $k \rightarrow k + 1$

A3. End

A4. Set $x = x_{k+1}$ and $f = F(x_{k+1})$

Bisection method

[x,f] = Bisection(F, x_L, x_R, Tol, $k_{\max}$)

A1. Set $k = 0$

A2. While $|x_R - x_L| > Tol$ and $k < k_{\max}$

- (i). Set $x_C = (x_L + x_R)/2$
- (ii). Evaluate $F(x_C)$
- (iii). If $|F(x_C)| < Tol$, Exit While.
- (iv). If $F(x_L)F(x_C) < 0$, set $x_R = x_C$. Otherwise set $x_L = x_C$.

A3. End

A4. Set $x = x_C$ and $f = F(x_C)$

(a) Identify three key differences between Newton's method and the bisection method. **[3 marks]**

Answer:

Any three of the following [1 mark each]:

- Bisection method converges with order 1, while Newton's method usually converges with order 2.
- Bisection method is robust and always finds a root in $[x_L, x_R]$ if $F(x_L)F(x_R) < 0$, whereas Newton's method might diverge if the initial guess x_0 is not close enough.
- Bisection method does not use derivatives, while Newton's method does.
- The stopping criteria of the two methods are different.
- Newton's method can be extended to vector-valued problems, while the bisection method only applies to scalar problems.

(b) Consider the case $F(x) = x^3 - x^2$.

(i) Find the exact solutions of $F(x^*) = 0$ by polynomial factorisation. **[3 marks]**

Answer:
We factorise $F(x) = x^3 - x^2 = x^2(x - 1) = 0$.
The solutions are $x = 1$ [1 mark] and $x = 0$ (double root) [2 marks].

(ii) For which of these solutions x^* is the convergence of Newton's method *quadratic* (with order 2) and for which is it only *linear* (with order 1)? Justify your answer by studying the properties of $F(x)$ at x^*. **[3 marks]**

Answer:
Newton's method applied to a polynomial converges quadratically to a root x^* if $F'(x^*) \neq 0$ (i.e. the root is simple) [1 mark].
Since $F'(x) = 3x^2 - 2x$, we note that $F'(0) = 0$ and $F'(1) = 1$, so that the method will converge quadratically at $x = 1$ [1 mark] and linearly at $x = 0$ [1 mark].

(iii) Is the initial bracket $[x_L, x_R] = [-1, 2]$ for the bisection method suitable for finding a solution $F(x^*) = 0$? Justify your answer by studying the properties of $F(x)$. **[2 marks]**

Answer:
We must check that $F(x_L)F(x_R) < 0$ [1 mark].

Since $F(-1) = (-1)^3 - (-1)^2 = -2$ and $F(2) = 2^3 - 2^2 = 4$, the necessary condition for applying the bisection method is satisfied [1 mark].

(iv) Apply one step of the bisection method starting from the initial bracket $[x_L, x_R] = [-1, 2]$. Which solution x^* does the method converge to? **[3 marks]**

Answer:
Initial bracket: $[x_L, x_R] = [-1, 2]$, $F(x_L) = -2$, $F(x_R) = 4$.
First midpoint: $x_C = (2 - 1)/2 = 0.5$, $F(x_C) = 0.5^3 - 0.5^2 = -0.125$ [1 mark].
Bracket condition: $F(x_C)F(x_R) < 0$, so we set $x_L = x_C = 0.5$ [1 mark].
Since the only root in the bracket $[x_L, x_R] = [0.5, 2]$ is $x = 1$, the bisection method must converge to that root [1 mark].

(v) Can the bisection method converge to a different solution $x^* \in [-1, 2]$ if we change the initial bracket? Justify your answer by studying the properties of $F(x)$ at x^*. **[3 marks]**

Answer:
The function $F(x)$ is locally concave and non-positive in the neighborhood of $x = 0$, which can be verified from $F(0) = F'(0) = 0$ and $F''(0) = -2$ (alternatively: they may plot the graph of the function and observe the same thing qualitatively). [1 mark].

Consequently, the bracket condition $F(x_L)F(x_R) < 0$ will fail in any sufficiently small neighborhood of $x = 0$ and the bisection method will generally not find this solution [1 mark].

The only exception are brackets for which the midpoint lands precisely on $x = 0$ by symmetry, e.g. the choice $[x_L, x_R] = [-2, 2]$ [1 mark]

(c) Explain how Newton's method can be modified to eliminate the requirement of computing the exact derivative $F'(x)$ (the *secant method*) How does this change the way the method has to be started? **[3 marks]**

Answer:
We use the approximation $F'(x_k) \approx (F(x_k) - F(x_{k-1})/(x_k - x_{k-1})$ [1 mark] to replace the derivative, resulting in the secant step [1 mark]

$$x_{k+1} = x_k - \frac{x_k - x_{k-1}}{F(x_k) - F(x_{k-1})} F(x_k).$$

We now need to provide two initial iterates, x_0 and x_{-1} to get started [1 mark].

[question 1 total: 20 marks]

CONTINUED ON FACING PAGE

Question 2

A scalar function $u(x,t)$ satisfies the following nonlinear partial differential equation

$$\frac{\partial u}{\partial t} = \left[\frac{\partial u}{\partial x}\right]^2 + 2u\frac{\partial^2 u}{\partial x^2}, \tag{1}$$

for $x \in [0,1]$ with boundary conditions $u(0,t)=\alpha$, $u(1,t)=\beta$, and $t>0$ with initial conditions $u(x,0)=U_0(x)$.

On a uniform grid of m nodes, with nodal spacing h, covering the domain $x \in [0,1]$, we first discretise the equation in space using the finite difference schemes

$$\frac{\partial u}{\partial x}(x_i) \approx \frac{u_{i+1}-u_i}{h}, \quad \frac{\partial^2 u}{\partial x^2}(x_i) \approx \frac{u_{i+1}-2u_i+u_{i-1}}{h^2}.$$

The semi-discrete problem $\dot{u}_i = f(u_{i-1},u_i,u_{i+1})$ is then discretised using backward Euler

$$\frac{u^{k+1}-u^k}{\Delta t} = f(u_{i-1}^{k+1}, u_i^{k+1}, u_{i+1}^{k+1})$$

where $\Delta t > 0$ is a constant time-step size.

(a) Show that the fully-discrete form of the problem may be written as:

$$F_i(\mathbf{U}) = \frac{u_i^{k+1}-u_i^k}{\Delta t} - \frac{[u_{i+1}^{k+1}]^2 - 3[u_i^{k+1}]^2 + 2u_i^{k+1}u_{i-1}^{k+1}}{h^2} = 0 \tag{2}$$

for $i=1,2,\ldots,m$, where $\mathbf{U}=(u_1^{k+1},u_2^{k+1},\ldots,u_m^{k+1})$ is the discrete solution vector at the next time step. **[7 marks]**

Answer:
The semi-discrete form is obtained by substituting the finite-difference approximations in space to the equation (1):

$$\frac{\partial u_i}{\partial t} = \tfrac{1}{h^2}\left[(u_{i+1}-u_i)^2 + 2u_i(u_{i+1}-2u_i+u_{i-1})\right] \quad \text{[1 mark]} \tag{3}$$

$$= \tfrac{1}{h^2}\left(u_{i+1}^2 - \cancel{2u_{i+1}u_i} + u_i^2 + \cancel{2u_iu_{i+1}} - 4u_i^2 + 2u_iu_{i-1}\right) \quad \text{[2 mark]} \tag{4}$$

$$= \tfrac{1}{h^2}\left(u_{i+1}^2 - 3u_i^2 + 2u_iu_{i-1}\right) \quad \text{[1 mark].} \tag{5}$$

The fully-discrete form is obtained using backward Euler as:

$$\frac{u_i^{k+1}-u_i^k}{\Delta t} = \frac{1}{h^2}\left([u_{i+1}^{k+1}]^2 - 3[u_i^{k+1}]^2 + 2u_i^{k+1}u_{i-1}^{k+1}\right). \text{ [2 marks]}$$

The nonlinear equation is obtained by moving all the terms to the left-hand side:

$$F_i = \frac{u_i^{k+1}-u_i^k}{\Delta t} - \frac{1}{h^2}\left([u_{i+1}^{k+1}]^2 - 3[u_i^{k+1}]^2 + 2u_i^{k+1}u_{i-1}^{k+1}\right) = 0. \text{ [1 mark]}$$

(b) The Jacobian matrix $\mathbf{J}$ for the nonlinear system $\mathbf{F}(\mathbf{U}) = \mathbf{0}$ is defined elementwise as

$$J_{ij} = \frac{\partial F_i}{\partial U_j}.$$

Show that the Jacobian for the discrete problem (2) is a tridiagonal matrix by computing the nonzero elements J_{ij} of the i'th row. **[4 marks]**

Answer:
For the function F_i we have:

$$J_{i,i-1} = -\frac{2U_i}{h^2} \text{ [1 mark]}$$

$$J_{i,i} = \frac{1}{\Delta t} - \frac{U_{i-1} - 3U_i}{2h^2} \text{ [1 mark]}$$

$$J_{i,i+1} = -\frac{2U_{i+1}}{h^2} \text{ [1 mark]}$$

All other elements $J_{i,\cdot}$ are zero. Therefore, the matrix is tridiagonal. [1 mark]

(c) Consider a numerical Jacobian approximation

$$J_{ij} \approx \frac{F_i(\mathbf{U} + \delta \mathbf{e}_j) - F_i(\mathbf{U})}{\delta}$$

where $\delta > 0$ is a small perturbation and $\mathbf{e}_j = (0, 0, \ldots, 1, 0, \ldots)^T$ is a vector with the j'th component equal to 1 and all others equal to 0.

Describe an efficient method for evaluating the numerical Jacobian of a tridiagonal matrix using only three evaluations of $\mathbf{F}$. **[2 marks]**

Answer:
Each column of the Jacobian has only 3 non-zero entries hence we can form 3 non-overlapping sets of Jacobian columns $(j_1, j_4, ...)$, $(j_2, j_5, ...)$, $(j_3, j_6, ...)$ [1 mark].

This allows us to perturb several variables simultaneously and evaluate the entire Jacobian with only 3 function calls [1 mark].

(d) Formulate in pseudocode an algorithm for solving the fully-discrete problem (2) using Newton method's in each time step, including:

- a time-stepping procedure that calls Newton's method at each time step;
- a choice of the initial guess for Newton's method at each iteration;
- an efficient solution of the linear system at each Newton iteration.

(Note: It is not necessary to provide pseudocode for solving the tridiagonal linear system, simply mention which algorithm you would choose.) **[7 marks]**

Answer:
Given the initial time t_0, the final time t_f and the time-step size dt:

% Time-stepping loop
Set U_0 equal to the initial condition. [1 mark]

For t = t_0 : dt : t_f
    Solve U = Newton(F, U_0, dt, tol, $maxIter$); [1 mark]
    Set $U_0 = U$;
End

% Newton iteration
Function U = Newton(F, U_0, dt, tol, $maxIter$)
    Set $k = 1$
    While $|F(U_{k-1})| > tol$ and $k < maxIter$ [1 marks]
        Evaluate Jacobian $J = dF(U_{k-1}, U_0, dt)$ [1 mark]
        Solve $J\delta = -F(U_{k-1})$ using the Thomas algorithm [2 marks]
        Update $U_k = U_{k-1} + \delta$ (add line-search if necessary) [1 mark]
        Set $k \rightarrow k + 1$
    End
End

[question 2 total: 20 marks]

Question 3

Provide brief answers to the following questions on computational linear algebra:

(a) Define what we mean when we say that a matrix A of size $n \times n$ is *sparse*. **[1 mark]**

Answer: A sparse $n \times n$ matrix has at most nz non-zero entries, where $nz \ll n^2$ and typically $nz = O(n)$. [1 mark]

Identify three different data structures for efficiently storing sparse matrices. **[3 marks]**

Answer:

- coordinate format [1 mark]
- row-major format [1 mark]
- column-major format [1 mark]

(b) Explain how the *LU factorisation* of an $n \times n$ matrix, $A = LU$, can be used to solve a linear system $Ax = b$. **[2 marks]**

Answer: Using the LU factors, we divide the problem $Ax = b$ into two sub-problems [1 mark]:

(i) $Lz = b$

(ii) $Ux = z$

The sub-problems can be solved efficiently by forward/backward-substitution because L and U are lower- and upper-triangular, respectively. [1 mark].

Explain why reordering the rows/columns of a sparse matrix A can increase the computational efficiency of LU factorisation. **[2 marks]**

Answer: Reordering the rows/columns of the matrix A can reduce its bandwidth [1 mark] and consequently reduce the fill-in of the LU factors L and U [1 mark].

(c) Describe the *greedy minimum degree algoritm* for reordering a sparse matrix A. **[4 marks]**

Answer:

(i) Extract the graph structure of the sparse matrix. [1 mark]

(ii) Calculate the degree of each node i in the graph. [1 mark]

(iii) Pick any starting node and renumber it as node 1.

(iv) For each renumbered node:
   - Order the non-renumbered neighbours of that node by their degree. [1 mark]
   - Renumber each of them in that sequence. [1 mark].

(d) Formulate in pseudocode the *Jacobi iteration* as an iterative method for solving the linear problem $Ax = b$. **[4 marks]**

Answer: Given an initial guess x_0 [1 mark], a stopping tolerance tol and maximum number of iterations $k_{\max}$:

(i) Set $k = 0$

(ii) Do

(iii) Compute current residual $r_k = b - Ax_k$ [1 mark]

(iv) Update $x_{k+1} = x_k + \text{diag}(A)^{-1} r_k$ [1 mark]

(v) Update $k \to k + 1$

(vi) While $\|r_k\| > tol$ and $k < k_{\max}$ [1 mark]

(e) Identify four properties of a good *preconditioner* M for a linear problem $Ax = b$ that allow the preconditioned problem $(M^{-1}A)x = M^{-1}b$ to be solved more efficiently using an iterative method (such as the conjugate gradient method) than the original problem. **[4 marks]**

Answer: Any four of the following [1 mark each]:

- The matrix M should be computationally cheap to assemble.
- The linear system $Mz = r$ should be computationally cheap to solve.
- The inverse M^{-1} should approximate the inverse A^{-1} in some sense.
- The preconditioned system $(M^{-1}A)x = M^{-1}b$ should require fewer iterations.
- The preconditioner M should be chosen based on the properties of A.
- The preconditioned system should be better conditioned, i.e. $\kappa(M^{-1}A) \ll \kappa(A)$.

[question 3 total: 20 marks]

[grand total: 60 marks]