Instructions

- You may rip off the last two pages as soon as you sit down.
- There are 34 marks available. It will be marked out of 30.
- No aides.
- Turn off all electronic media and store them under your desk.
- You may ask only one question during the examination: "May I go to the washroom?"
- Asking any other question will result in a deduction of 5 marks from the exam grade.
- If you think a question is ambiguous, write down your assumptions and continue.
- Do not leave during first hour or after there are only 15 minutes left.
- Do not stand up until all exams have been picked up.
- There are questions on both sides of the pages.
- If a question only asks for an answer, you do not have to show your work to get full marks; however, if your answer is wrong and no rough work is presented to show your steps, no part marks will be awarded.
- Answer the questions in the spaces provided. If you require additional space to answer a question, please use the provided blank page and refer to this page in your solutions.

1. [**3**] Multiply the following two numbers shown in the double-precision floating-point representation:

   ```
   bff6000000000000    1 01111111111 011000...0
   3ff8000000000000    0 01111111111 100000...0
   ```

   Each row is the same number, only the first is in the hexadecimal representaiton, and the second is in the binary representation. You may give your answer in either hexadecimal or in binary, as you wish.

2. [**3**] To approximate the derivative of $\sin(x)$ at $x = 1$ with a value of $h = 0.001$, we would preform the following calculation:

$$\begin{aligned}\frac{\sin(1.000) - \sin(0.9999)}{0.0001} &= \frac{0.84147098\cdots - 0.84141695\cdots}{0.0001} \\ &= 0.54034437\cdots\end{aligned}$$

   which is a reasonable approximation of $\cos(1) = 0.54030230\cdots$. Show how this fails to give a precise result if we were to perform these calculations by rounding each value to only four decimal digits of precision (as with our six decimal-digit floating point representation) at each step. The dots $\cdots$ are used to represent digits in the mantissa that are not being shown.

   What phenomenon described in class is this an example of? Either give the title of the phenomenon or describe the phenomenon.

3. [**4**] Show that the error of the approximation of the second derivative

$$f^{(2)}(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$$

   is equal to $-\frac{1}{12}f^{(4)}(\xi)h^2$ where $x - h \leq \xi \leq x + h$. You must show and explain each step in your calculations. You should use 3rd-order Taylor series for $f(x+h)$ and $f(x-h)$. Show where you use the intermediate-value theorem.

4. **[1]** If you consider the equation $x = 3.2x(1 - x)$ and try to find a solution using fixed-point iteration, after a hundred iterations, you note that your iterations jump between two values:

$$\ldots, 0.5130445095326300, 0.7994554904673700, 0.5130445095326300, 0.7994554904673700, \ldots$$

Justify whether or not these are the two solutions to the given quadratic equation. Note that $\frac{3.2-1}{3.2} = 0.6875$

5. **[4]** Use Gaussian elimination with partial pivoting to reduce the following augmented matrix to row-echelon form.

$$\left(\begin{array}{rrr|r} 2.1 & 1.1 & 3.0 & 8.9 \\ -7.0 & 3.0 & 5.0 & 2.0 \\ -2.8 & 5.2 & 7.0 & 7.8 \end{array}\right)$$

You do not have to find the solution to this system of linear equations (that is, you do not have to apply backward substitution); however, if you wish to check your answer, the solution is $\begin{pmatrix} 1 \\ -2 \\ 3 \end{pmatrix}$.

6. **[4]** Suppose you are dealing with a real-time system where you are periodically sampling the value of a sensitive sensor ten times per second (so 10Hz); however, there is a reasonable probability that once in any ten seconds, the reading may become lost (you know when a reading is lost). Thus, in estimating the derivative at the current time, it may no longer be possible to use the $O(h^2)$ formula

$$y^{(1)}(t_k) \approx \frac{3y_k - 4y_{k-1} + y_{k-2}}{2h}$$

as one of y_k, y_{k-1} or y_{k-2} may be lost.

You could fall back on the $O(h)$ approximations such as

$$y^{(1)}(t_k) \approx \frac{y_k - y_{k-1}}{h}, y^{(1)}(t_k) \approx \frac{y_{k-1} - y_{k-2}}{h} \text{ or } y^{(1)}(t_k) \approx \frac{y_k - y_{k-2}}{2h}$$

depending on which value is missing, but an analysis shows that you would be forced to increase the sample rate to 25Hz to ensure sufficient precision.

Without actually finding the formulas, describe what steps you would take to find an approximation of the derivative at t_k that is more likely to be $O(h^2)$ than the formulas described above. You may use previous readings. You do not have to prove your approach will guarantee that the error is $O(h^2)$.

7. **[3]** Assume that we want to apply the trapezoidal formula for approximating the integral from a to b by breaking the interval into n equally-sized sub-intervals where $x_k = a + hk$ where $h = \frac{b-a}{n}$. To do the error analysis, we must sum all the errors to get

$$\sum_{k=1}^{n} -\frac{1}{12} f^{(2)}(\xi_k) h^3$$

First, describe the bound on each of the ξ_k, and then, second, show how this formula can be simplified to

$$-\frac{(b-a)}{12} f^{(2)}(\xi) h^2$$

and describe how we know that $a \leq \xi \leq b$.

8. **[1]** A matrix is more numerically stable if the condition number is [very large] or [closer to one] (circle one).

9. [**1**] Circle which of the following properties are still true when using double-precision floating-point computations?

   (a) $x + y = y + x$

   (b) $x + (y + z) = (x + y) + z$

   (c) $x + y = x$ if and only if $y = 0$

   (d) $(x + y) - x = y$

   Note that all of these properties are true if you are using real numbers. It is the fixed precision of the double-precision floating point representation that causes some of these properties to no longer hold.

10. [**3**] Apply one step of the bracketed secant method to find a better approximation of the root of $x^3 - 0.01$ given that you know the root is on the interval $[0, 1]$. Which end-point will you update?

11. [**2**] Suppose you have found the least-squares best-fitting quadratic that passes through the last n periodically-sampled points. Suppose that this polynomial is $at^2 + bt + c$ (where b is negative) and everything was scaled and shifted, as described in the course. How would you determine if this polynomial has a root on the next time step, and if you determine it has a root, which formula would you use to estimate how far into the time step that root occurs? You can use either $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ or $\frac{-2c}{b \pm \sqrt{b^2 - 4ac}}$, but which will you choose and do you use $+$ or $-$ when you are given the choice of $\pm$?

12. [**2**] Does the method of successive over-relaxation rely on a weighted average of the two previous approximations x_k and x_{k-1}? If not, explain clearly why not, and if so, for what values of ω is this a convex combination?

13. **[3]** Write down the system of linear equations that must be solved to find $\Delta\mathbf{u}_0$ when trying to find the simultaneous solution of the system of three non-linear equations

$$xyz = 1, xy + yz = 1, \text{ and } x + y + z = 1$$

starting with the initial guess $\mathbf{u}_0 = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$. Do not take any steps towards trying to solve this system of linear equations.

YOU MAY RIP THESE LAST TWO PAGES OFF
USE THIS PAGE IF ADDITIONAL SPACE IS REQUIRED
Clearly state the question number being answered and refer the marker to this page.
If this page is ripped off, you do not have to staple it back together with your other pages.

Floating-point representations: $\pm$`EENMMM` represents $\pm$`N.MMM` $\times 10^{\texttt{EE}-49}$ and the 64 bits `seeeeeeeeeeebbbbbb`$\cdots$`b` represents

$$(-1)^{\texttt{s}}\texttt{1.bbbbbb}\cdots\texttt{b} \times 2^{\texttt{eeeeeeeeeee}-01111111111}$$

where `0b01111111111` $= 1023 =$ `0x3ff`. Recall 1 is `+491000` or `0x3ff0000000000000`.
Fixed-point theorem: To approximate a solution to $x = f(x)$, choose x_0 and let $x_k \leftarrow f(x_{k-1})$.
Gaussian elimination with partial pivoting: This is the Gaussian elimination algorithm but always swapping appropriate rows so that the largest entry is in the pivot position (the row that will be used to eliminate that term in all subsequent rows).
***n*th-order Taylor series:** If h is small, expanding around x yields:

$$f(x+h) = \left(\sum_{k=0}^{n} \frac{1}{k!} f^{(k)}(x) h^k\right) + \frac{1}{(n+1)!} f^{(n+1)}(\xi) h^{n+1}$$

where $x \leq \xi \leq x+h$. Otherwise, if x is close to x_0, expanding around x_0 yields:

$$f(x) = \left(\sum_{k=0}^{n} \frac{1}{k!} f^{(k)}(x_0) (x-x_0)^k\right) + \frac{1}{(n+1)!} f^{(n+1)}(\xi) (x-x_0)^{n+1}$$

where $x_0 \leq \xi \leq x$.

```
double horner( double a[], unsigned int const degree, double const x ) {
    // The coefficient of xˆk is a[k]
    double result{ a[degree] };

    for ( std::size_t k{degree - 1}; k < degree; --k ) {
        result = result*x + a[k];
    }

    return result;
}
```

Noise: Averaging noisy values with zero bias mitigates the effect, while differentiating noisy values magnifies the effect. Use interpolating polynomials if the data is accurate and precise, but use least squares best-fitting polynomials if the data is accurate but not precise (that is, the data has significant noise). If the data is not accurate, we cannot recover the underlying signal.
Evaluating interpolating polynomials: For interpolating between t_k and t_{k-1} where t_k is the time of the most recent data point, shift and scale to $\ldots, -2.5, -1.5, -0.5$ and 0.5 to ensure that $-0.5 < \delta < 0.5$ to evaluate the polynomial at the point $\frac{t_{k-1}+t_k}{2} + \delta h$ where h is the time step between readings. Note, you do not have to know these formulas explicitly; rather, you must understand the idea behind deriving these. For example, why to we shift and scale so that our choice of δ is such that $|\delta| < 0.5$.
Derivatives:
Centered three-point:

$$f^{(1)}(x) = f^{(1)}(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{1}{6} f^{(3)}(\xi) h^2$$

Backward two-point:

$$y^{(1)}(t) = \frac{y(t) - y(t-h)}{h} + \frac{1}{2} y^{(2)}(\tau) h$$

Backward three-point:

$$y^{(1)}(t) = \frac{3y(t) - 4y(t-h) + y(t-2h)}{2h} + \frac{1}{3} y^{(3)}(t) h^2 + \mathrm{O}\left(h^3\right)$$

Second derivatives:
Centered three-point:

$$f^{(2)}(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} - \frac{1}{12} f^{(4)}(\xi)\, h^2$$

Backward three-point:

$$y^{(2)}(t) = \frac{y(t) - 2y(t-h) + y(t-2h)}{h^2} + y^{(3)}(\tau)\, h$$

Backward four-point:

$$y^{(2)}(t) = \frac{2y(t) - 5y(t-h) + 4y(t-2h) - y(t-3h)}{h^2} + \frac{11}{12} y^{(4)}(t)\, h^2 + \mathrm{O}\left(h^3\right)$$

Integrals:
Two-point (trapezoidal rule):

$$\int_{x_{k-1}}^{x_k} f(x)\,\mathrm{d}x = \left(\frac{1}{2} f(x_{k-1}) + \frac{1}{2} f(x_k)\right) h - \frac{1}{12} f^{(2)}(\xi)\, h^3$$

Centered four-point:

$$\int_{x_{k-1}}^{x_k} f(x)\,\mathrm{d}x = \left(-\frac{1}{24} f(x_{k-2}) + \frac{13}{24} f(x_{k-1}) + \frac{13}{24} f(x_k) - \frac{1}{24} f(x_{k+1})\right) h - \frac{11}{720} f^{(4)}(t_k)\, h^5 + \mathrm{O}\left(h^6\right)$$

Simpson's rule:

$$\int_{x_{k-1}}^{x_{k+1}} f(x)\,\mathrm{d}x = \left(\frac{1}{6} f(x_{k-1}) + \frac{4}{6} f(x_k) + \frac{1}{6} f(x_{k+1})\right) (2h) - \frac{1}{90} f^{(4)}(\xi)\, h^5$$

Backward three-point (half Simpson's rule):

$$\int_{t_{k-1}}^{t_k} y(t)\,\mathrm{d}x = \left(\frac{5}{12} y(t_k) + \frac{8}{12} y(t_{k-1}) - \frac{1}{12} y(t_{k-2})\right) h - \frac{1}{24} y^{(3)}(t_k)\, h^4 + \mathrm{O}\left(h^5\right)$$

Backward four-point:

$$\int_{t_{k-1}}^{t_k} y(t)\,\mathrm{d}x = \left(\frac{9}{24} y(t_k) + \frac{19}{24} y(t_{k-1}) - \frac{5}{24} y(t_{k-2}) + \frac{1}{24} y(t_{k-3})\right) h + \frac{19}{720} y^{(4)}(t_k)\, h^5 + \mathrm{O}\left(h^6\right)$$

As Simpson's rule spans two time intervals, it is less useful, but it is interesting with its comparison with the trapezoidal rule applied twice versus one application of Simpson's rule.
Any integral formula can be applied repeatedly on the interval $[a, b]$ by dividing the interval into n equally-spaced sub-intervals of width $h = \frac{b-a}{n}$ and then setting $x_k = a + kh$ or $t_k = a + kh$.
Least squares: In general, if we want to find the best approximation of an n-dimensional vector $\mathbf{y}$ by a linear linear combination of m vectors $\mathbf{v}_1, \ldots, \mathbf{v}_m$ (where $m < n$), we create the matrix $V = (\mathbf{v}_1 \cdots \mathbf{v}_m)$ and solve $V^{\mathrm{T}} V \mathbf{a} = V^{\mathrm{T}} \mathbf{y}$. More specific to this course, having shifted and scaled the n most recent t-values onto $0, -1, -2, \ldots, -n+1$, with y values $\mathbf{y} = (y_k, y_{k-1}, y_{0-2}, \ldots, y_{k-n+1})$, we solve $V^{\mathrm{T}} V \mathbf{a} = V^{\mathrm{T}} \mathbf{y}$ for the coefficients of the least-squares best-fitting polynomial, generally of degree one (linear or $\alpha_1 t + \alpha_0$) or two (quadratic or $\alpha_2 t^2 + \alpha_1 t + \alpha_0$). We can find the $2 \times n$ or $3 \times n$ matrix to calculate $\mathbf{a} = \left(V^{\mathrm{T}} V\right)^{-1} V^{\mathrm{T}} \mathbf{y}$.

Value being estimated	Linear estimation
$y(t_k)$	α_0
$y(t_k + h)$	$\alpha_0 + \alpha_1$
$y^{(1)}(t_k)$	α_1/h
$\int_{t_k-h}^{t_k} y(t)\mathrm{d}t$	$(\alpha_0 - \alpha_1/2)h$
$\int_{t_k}^{t_k+h} y(t)\mathrm{d}t$	$(\alpha_0 + \alpha_1/2)h$

Value being estimated	Quadratic estimation
$y(t_k)$	α_0
$y(t_k + h)$	$\alpha_0 + \alpha_1 + \alpha_2$
$y^{(1)}(t_k)$	α_1/h
$y^{(2)}(t_k)$	$2\alpha_2/h^2$
$\int_{t_k-h}^{t_k} y(t)\mathrm{d}t$	$(\alpha_0 - \alpha_1/2 + \alpha_2/3)h$
$\int_{t_k}^{t_k+h} y(t)\mathrm{d}t$	$(\alpha_0 + \alpha_1/2 + \alpha_2/3)h$

References to both binary search and interpolation search are not applicable to this course. Instead, they are introduced into the course to demonstrate parallels between the binary search and bisection method, and interpolation search and the bracketed secant method.

Bisection: Given an interval $[a, b]$ with $f(a)$ and $f(b)$ having opposite signs, let $c \leftarrow \frac{a+b}{2}$ and update whichever endpoint has the same sign as $f(c)$. $O(h)$.

Bracketed secant: Given an interval $[a, b]$ with $f(a)$ and $f(b)$ having opposite signs, let $c \leftarrow \frac{af(b)-bf(a)}{f(b)-f(a)}$ and update whichever endpoint has the same sign as $f(c)$. $O(h)$.

Secant: Given two initial approximations x_0 and x_1 with $|f(x_0)| > |f(x_1)|$, let $x_2 \leftarrow \frac{x_0 f(x_1)-x_1 f(x_0)}{f(x_1)-f(x_0)}$. $O(h^{1.618})$.

Muller's: Given three initial approximations, interpolate $(x_0 - x_2, y_0)$, $(x_1 - x_2, y_1)$ and $0, y_2)$ and find the smaller root of the interpolating quadratic, call this δ and set $x_3 = x_2 + \delta$. $O(h^{1.839})$.

Inverse quadratic interpolation: Given three initial approximations, interpolate (y_0, x_0), (y_1, x_1) and (y_2, x_2) and find let x_3 be the constant coefficient of this interpolating quadratic polynomial. $O(h^{1.839})$.

Newton's: Given an initial approximation x_0, let $x_1 \leftarrow x_0 - \frac{f(x_0)}{f^{(1)}(x_0)}$. $O(h^2)$.

Fixed-point iteration for systems of linear equations: Given a square $n \times n$ matrix A, if D_A is the matrix corresponding to the diagonal entries of A, and A_{off} consists of all the off-diagonal entries of A (so $A = D_A + A_{\text{off}}$), the we can rewrite $A\mathbf{x} = \mathbf{b}$ as the equation $\mathbf{x} = D_A^{-1}(\mathbf{b} - A_{\text{off}}\mathbf{x})$. Let the entries of $\mathbf{x}_k = (x_{k,1}, x_{k,2}, ..., x_{k,n})$.

Jacobi: For i from 1 to n, set $x_{k,i} \leftarrow D_A^{-1}(\mathbf{b} - A_{\text{off}}\mathbf{x}_{k-1})$

Gauss-Seidel: Set $\mathbf{x}_k \leftarrow \mathbf{x}_{k-1}$, and then for i from 1 to n, update $x_{k,i} \leftarrow D_A^{-1}(\mathbf{b} - A_{\text{off}}\mathbf{x}_k)$

Successive over-relaxation: Given the approximation x_{k-1} and having iterated an algorithm once more to get the approximation x_k, we can move an addition $100\omega\%$ in the direction of the newer approximation by updating $x_k \leftarrow (1 + \omega)x_k - \omega x_{k-1}$.

Newton's method in two dimensions: Given functions $f(x, y)$ and $g(x, y)$ and an approximation to a simultaneous root (x_k, y_k), we can solve

$$\begin{pmatrix} \frac{\partial}{\partial x} f(x_k, y_k) & \frac{\partial}{\partial y} f(x_k, y_k) \\ \frac{\partial}{\partial x} g(x_k, y_k) & \frac{\partial}{\partial y} g(x_k, y_k) \end{pmatrix} \begin{pmatrix} \Delta x_k \\ \Delta y_k \end{pmatrix} = \begin{pmatrix} -f(x_k, y_k) \\ -g(x_k, y_k) \end{pmatrix}$$

and then let $x_{k+1} \leftarrow x_k + \Delta x_k$ and $y_{k+1} \leftarrow y_k + \Delta y_k$.

Newton's method in n dimensions: More generally, approximating $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, given an approximation $\mathbf{x}_k$, solve $\mathbf{J}(\mathbf{f})(\mathbf{x}_k)\Delta\mathbf{x}_k = -\mathbf{f}(\mathbf{x}_k)$ and then let $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \Delta\mathbf{x}_k$.