MTH 4422 Final Exam Study Guide

FALL 2017

Pat Rossi Name _____

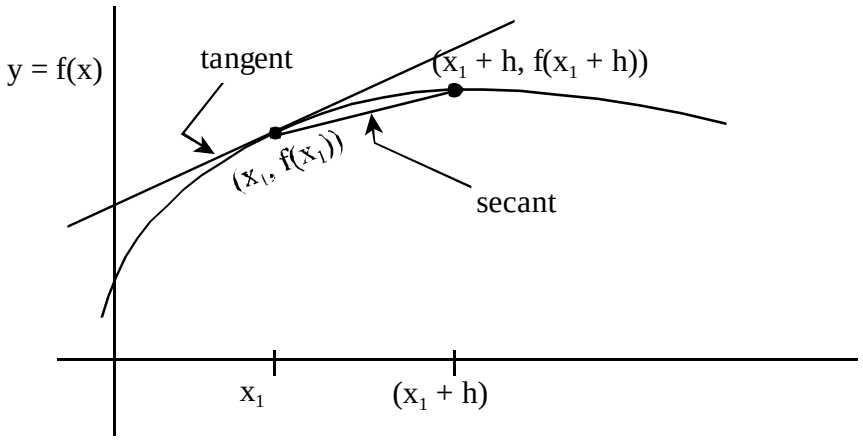
Instructions. Answer the following questions thoroughly.

1. Explain how the derivative of $f(x)$ is computed numerically, using either the Forward Difference Method, or the Central Difference Method.

   Forward Difference: $f'(x_1)$ is the slope of the line tangent to the graph of $f(x)$ at the point $(x_1, f(x_1))$. If we go h units to the right of x_1 and plot the corresponding point on the graph of $f(x)$, this point and the point $(x_1, f(x_1))$ can be used to draw a secant. The slope of this secant is approximately the same as the slope of the line tangent, $f'(x_1)$ (see below). This yields the approximation:

   $$f'(x_1) \approx \frac{f(x_1 + h) - f(x_1)}{h}.$$

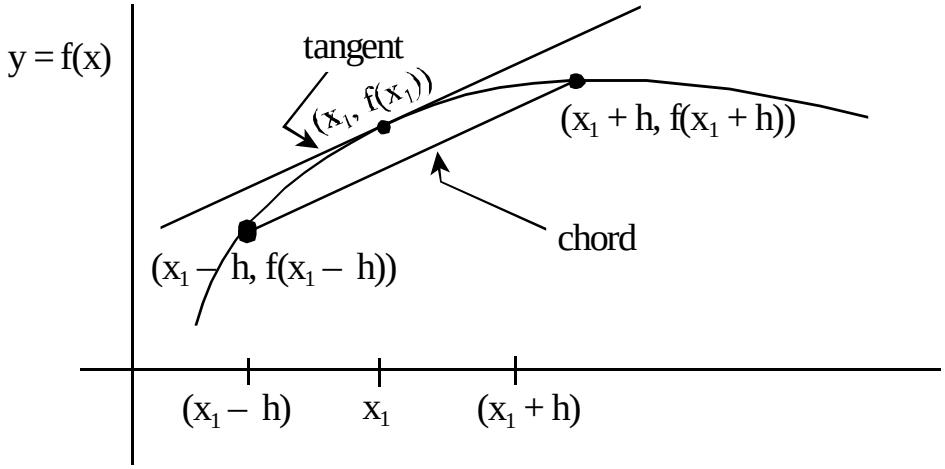
   Furthermore, this approximation improves with smaller values of h. To simulate the process of letting $h \to 0$, we cut h in half each iteration, and stop when two successive approximations are within the desired tolerance. The last approximation is our answer.



Central Difference: $f'(x_1)$ is the slope of the line tangent to the graph of $f(x)$ at the point $(x_1, f(x_1))$. If we go h units in either direction of x_1 and plot the corresponding points on the graph of $f(x)$, these points can be used to draw a chord with endpoints $(x_1 - h, f(x_1 - h))$ and $(x_1 + h, f(x_1 + h))$. The slope of this chord is approximately the same as the slope of the line tangent, $f'(x_1)$ (see below). This yields the approximation:

$$f'(x_1) \approx \frac{f(x_1 + h) - f(x_1 - h)}{2h}.$$

Furthermore, this approximation improves with smaller values of h. To simulate the process of letting $h \to 0$, we cut h in half each iteration, and stop when two successive approximations are within the desired tolerance. The last approximation is our answer.



2. Explain how Taylor's Series can be used to approximate derivatives of higher order. In particular, explain how the second derivative of a function $f(x)$ can be approximated.

   Consider the Taylor Series expansion for $f(x)$ with center x_0 :

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + f''(x_0)\frac{(x - x_0)^2}{2} + \dots$$

   If we define $h = x - x_0$, and hence, $x = x_0 + h$, we have:

$$f(x_0 + h) = f(x_0) + f'(x_0)h + f''(x_0)\frac{h^2}{2} + \dots$$

   For some number ψ_1 with $0 < \psi_1 < h$, we can write this as:

$$f(x_0 + h) = f(x_0) + f'(x_0)h + f''(x_0)\frac{h^2}{2} + f'''(x_0)\frac{h^3}{3!} + f^{(4)}(\psi_1)\frac{h^4}{4!} \qquad (1)$$

   Similarly, for a number ψ_2, with $-h < \psi_2 < 0$, we have:

$$f(x_0 - h) = f(x_0) - f'(x_0)h + f''(x_0)\frac{h^2}{2} - f'''(x_0)\frac{h^3}{3!} + f^{(4)}(\psi_2)\frac{h^4}{4!}. \qquad (2)$$

   Adding equations 1 and 2 yields:

$$f(x_0 + h) + f(x_0 - h) = 2f(x_0) + f''(x_0)h^2 + f^{(4)}(\psi_1)\frac{h^4}{4!} + f^{(4)}(\psi_2)\frac{h^4}{4!}.$$

   Solving for $f''(x_0)$, we have:

$$f''(x_0) = \frac{1}{h^2}\left[f(x_0 + h) - 2f(x_0) + f(x_0 - h)\right] - h^2\left[\frac{f^{(4)}(\psi_1)}{4!} + \frac{f^{(4)}(\psi_2)}{4!}\right]$$

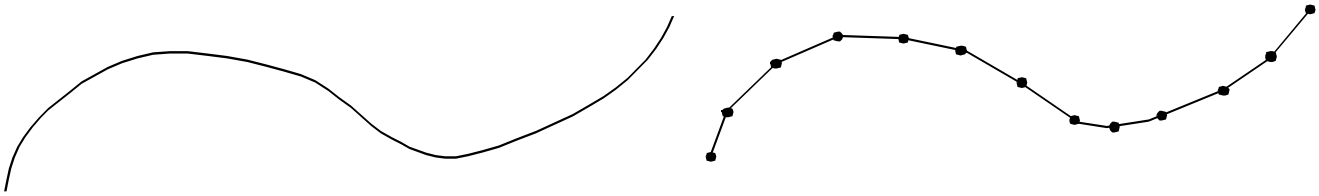
   Truncating the last term, we have:

$$f''(x_0) \approx \frac{1}{h^2}\left[f(x_0 + h) - 2f(x_0) + f(x_0 - h)\right],$$

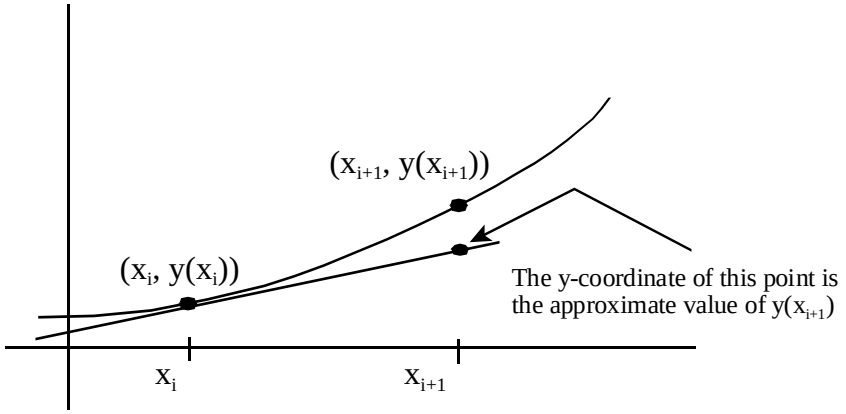
   with error of order h^2. This allows us to compute the second derivative in terms of the values of the function itself - something we might want to do if we don't know what our function $f(x)$ is, but just have the data values.

3. Describe and explain Euler's Method (the "Bow and Arrow Method"), for solving a first order linear differential equation, using a geometric approach.

   This algorithm is designed to approximate the solution of first order linear differential equations of the form: $y' = f(y, x)$, on the interval $[a, b]$, subject to initial condition $y(a) = \alpha$. (The solution of such an equation is the function y evaluated at $x = b$. (i.e. the solution is $y(b)$.)) Geometrically, Euler's method is based on the assumption that the graph of a function $y(x)$ can be approximated by many small line segments, as shown below.



   Given a known point on the graph of $y(x)$, let's say $(x_i, y(x_i))$, we can approximate the y-coordinate of another point $(x_{i+1}, y(x_{i+1}))$ close by on the graph, by travelling along the line tangent to the graph at the point $(x_i, y(x_i))$, until we reach the x-coordinate x_{i+1}. (See below.) We call this "shooting the arrow."



   If h is small, this will be a good approximation. How do we get the value of $y_{i+1} \approx y(x_{i+1})$? We use the slope formula: $m = \frac{y_2 - y_1}{x_2 - x_1}$. Rewritten in terms of our points $(x_i, y(x_i))$, and $(x_{i+1}, y(x_{i+1}))$, this becomes:

$$y'(x_i) = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}.$$

   (Recall, that $y'(x_i)$ is the slope of the line tangent.)

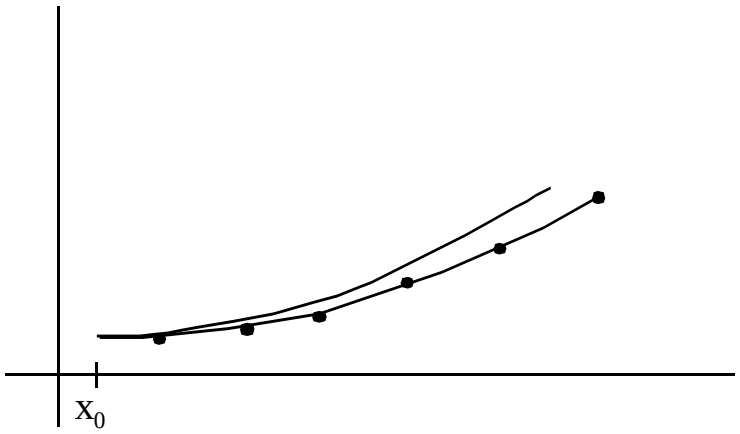
   $\Rightarrow y_{i+1} - y_i = y'(x_i)(x_{i+1} - x_i)$

   $\Rightarrow y_{i+1} = y_i + y'(x_i)(x_{i+1} - x_i)$

   Since $y = f(y, x)$, and $h = x_{i+1} - x_i$, this becomes:

$$y_{i+1} = y_i + f(y_i, x_i)\, h.$$

This equation can be used recursively to generate the values of y for each value of x_i.
Our solution is $y(b) \approx y_n$.
(The approximation to y is shown graphically, below)



4. Describe and explain Euler's Method (the "Bow and Arrow Method") using the Taylor's Series approach.

   This algorithm is designed to approximate the solution of first order linear differential equations of the form: $y' = f(y, x)$, on the interval $[a, b]$, subject to initial condition $y(a) = \alpha$. (The solution of such an equation is the function y, evaluated at the point $x = b$. (i.e., the solution is $y(b)$.)) To see how we derive an approximation for the function y at certain values of x, consider the Taylor series expansion of the function y about the point x_0 :

$$y(x) = y(x_0) + y'(x_0)(x - x_0) + y''(x_0)\frac{(x - x_0)^2}{2} + \ldots + y^{(n)}(x_0)\frac{(x - x_0)^n}{n!} + \ldots$$

   Letting $h = x - x_0$, this becomes:

$$y(x_0 + h) = y(x_0) + y'(x_0)h + y''(x_0)\frac{h^2}{2} + \ldots + y^{(n)}(x_0)\frac{h^n}{n!} + \ldots$$

   This can be expressed as a finite series:

$$y(x_0 + h) = y(x_0) + y'(x_0)h + y''(\psi)\frac{h^2}{2},$$

   where $0 < \psi < h$. Truncating the last (error) term, we have:

$$y(x_0 + h) \approx y(x_0) + y'(x_0)h. \tag{3}$$

   Partitioning the interval $[a, b]$ into n sub-intervals of equal length h, we define:
   $x_i = a + ih$, $x_{i+1} = x_i + h$, and $y_i = y(x_i)$.
   Proceeding inductively, equation 3 becomes:

$$y(x_i + h) \approx y(x_i) + y'(x_i)h,$$

   which can, in turn, be written as:

$$y_{i+1} \approx y_i + y'(x)h.$$

   But notice that since y is the solution of the equation $y' = f(y, x)$, this becomes:

$$y_{i+1} \approx y_i + f(y_i, x_i)h.$$

   Thus, we have a way of inductively generating the values of $y(x)$ for values of $x = x_1, x_2, \ldots, x_n$.
   Our solution is $y(b) \approx y_n$.

5. Explain the method used to compute the inverse of a square matrix (using a computer).

"By hand," the inverse of a matrix A is computed by annexing A with the identity matrix I, to form the so called augmented matrix. Forward elimination is performed on the augmented matrix, and then backward elimination is performed, transforming the left side of the augmented matrix into the identity I. If we were to do this by computer, we'd want to use "maximal pivoting" as we do forward elimination, so that we have as little "round off error" as possible. The problem is, we can't do "maximal pivoting" as we perform backward elimination, because the process of switching rows will change the left side of the augmented matrix into a matrix that is no longer "upper triangular." So THIS METHOD FOR FINDING AN INVERSE WON'T WORK.

An alternate method is to assume that the inverse exists. We'll call it B. We'll illustrate what happens with the 3×3 case:

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \underbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}}_{B} =$$

$$\underbrace{\begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} & a_{11}b_{13} + a_{12}b_{23} + a_{13}b_{33} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} & a_{21}b_{13} + a_{22}b_{23} + a_{23}b_{33} \\ a_{31}b_{11} + a_{32}b_{21} + a_{33}b_{31} & a_{31}b_{12} + a_{32}b_{22} + a_{33}b_{32} & a_{31}b_{13} + a_{32}b_{23} + a_{33}b_{33} \end{bmatrix}}_{AB}$$

i.e.,

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \underbrace{\begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} & a_{11}b_{13} + a_{12}b_{23} + a_{13}b_{33} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} & a_{21}b_{13} + a_{22}b_{23} + a_{23}b_{33} \\ a_{31}b_{11} + a_{32}b_{21} + a_{33}b_{31} & a_{31}b_{12} + a_{32}b_{22} + a_{33}b_{32} & a_{31}b_{13} + a_{32}b_{23} + a_{33}b_{33} \end{bmatrix}}_{AB}$$

Since the matrix I is equal to the matrix AB, each entry in AB must be equal to the corresponding entry in I.

Comparing the first column of the two matrices yields the system of equations:

$$\begin{aligned} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} &= 1 \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} &= 0 \\ a_{31}b_{11} + a_{32}b_{21} + a_{33}b_{31} &= 0 \end{aligned}$$

Comparing the second column of the two matrices yields the system of equations:

$$\begin{aligned} a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} &= 0 \\ a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} &= 1 \\ a_{31}b_{12} + a_{32}b_{22} + a_{33}b_{32} &= 0 \end{aligned}$$

Comparing the third column of the two matrices yields the system of equations:

$$\begin{aligned} a_{11}b_{13} + a_{12}b_{23} + a_{13}b_{33} &= 0 \\ a_{21}b_{13} + a_{22}b_{23} + a_{23}b_{33} &= 0 \\ a_{31}b_{13} + a_{32}b_{23} + a_{33}b_{33} &= 1 \end{aligned}$$

In matrix form these equations can be written as:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} b_{12} \\ b_{22} \\ b_{32} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} b_{13} \\ b_{23} \\ b_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

These systems can be solved using "forward elimination with maximal pivoting, and back substitution.." Thus we find the entries of matrix B, the inverse of A.

6. Explain how a system of n equations in n variables is solved using forward elimination, maximal pivoting, and back substitution.

   The system of equations

$$\begin{array}{ccccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & + & \dots & + & a_{1n}x_n & = & b_1 \\ a_{21}x_2 & + & a_{22}x_2 & + & a_{23}x_3 & + & \dots & + & a_{2n}x_n & = & b_2 \\ \vdots & & \vdots & & \vdots & & & & \vdots & & \vdots \\ a_{n1}x_1 & + & a_{n2}x_2 & + & a_{n3}x_3 & + & \dots & + & a_{nn}x_n & = & b_n \end{array}$$

   can be represented in matrix form as:

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix},$$

   and more compactly as:

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} & b_n \end{array}\right].$$

   Theoretically, we would solve this matrix equation by:

   (a) Performing *forward elimination* (eliminating below the main diagonal) and in the process, making the diagonal elements equal to 1

   (b) Performing *backward elimination* (eliminating above the main diagonal)

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} & b_n \end{array}\right] \Rightarrow \left[\begin{array}{cccc|c} 1 & a_{12}^* & \dots & a_{1n}^* & b_1^* \\ 0 & 1 & \dots & a_{2n}^* & b_2^* \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & 1 & b_n^* \end{array}\right] \Rightarrow \left[\begin{array}{cccc|c} 1 & 0 & \dots & 0 & b_1^* \\ 0 & 1 & \dots & 0 & b_2^* \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & 1 & b_n^* \end{array}\right]$$

   This equivalent to:

$$\left[\begin{array}{c} x_1 \\ x_2 \\ \vdots \\ x_n \end{array}\right] = \left|\begin{array}{c} b_1^* \\ b_2^* \\ \vdots \\ b_n^* \end{array}\right|,$$

   which is the solution of the system of equations.

When it comes to solving such a system of equations by computer, however, there's a problem – round-off error. Severe round-off error occurs when we divide a large number by a small number (i.e., numerator is several orders of magnitude larger than the denominator) and then propagate and magnify this error through numerous ensuing computations.

When solving a system of a computations numerically (i.e. using a computer), we can effectively minimize such round-off error during forward elimination, using a scheme called **maximal pivoting.**

Here's how maximal pivoting is employed:

Recall that when eliminating below the main diagonal in column j (for $j = 1$ to $(n-1)$), we replace row i with the sum of itself plus $\left(-\frac{a_{ij}}{a_{jj}}\right) \cdot row\ j$, for $i = (j+1)$ to n. We run the risk of creating large round-off error if a_{ij} is significantly larger then a_{jj}. To prevent this from happening, we compare all entries in column j, which lie on or below the main diagonal. Let's say that row k is the row having the largest entry (magnitude-wise) in column j. Then, before eliminating the entries in column j which lie below the main diagonal, we switch rows j and k.

This assures us that when we eliminate the entries lying below the main diagonal in column j, by replacing row i with the sum of itself plus $\left(-\frac{a_{ij}}{a_{jj}}\right) \cdot row\ j$, the denominator a_{jj} is always at least as large as the numerator a_{ij}. This is called **maximal pivoting.**

Thus, using forward elimination with maximal pivoting, we transform the original system to the "upper triangular" system show below:

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} & b_n \end{array}\right] \Rightarrow \left[\begin{array}{cccc|c} a_{11}^* & a_{12}^* & \dots & a_{1n}^* & b_1^* \\ 0 & a_{22}^* & \dots & a_{2n}^* & b_2^* \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & a_{nn}^* & b_n^* \end{array}\right]$$

At this point, we might be tempted to perform backward elimination. The only problem is that if we do, then we can't employ maximal pivoting techniques as we did in forward elimination. The reason is that in order to employ maximal pivoting, we must switch rows. But rows can't be switched during backward elimination, otherwise the matrix won't remain "upper triangular." (shown below)

$$\left[\begin{array}{cccc|c} a_{11}^* & a_{12}^* & \dots & a_{1n}^* & b_1^* \\ 0 & a_{22}^* & \dots & a_{2n}^* & b_2^* \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & a_{nn}^* & b_n^* \end{array}\right] \Rightarrow \left[\begin{array}{cccc|c} a_{11}^* & a_{12}^* & \dots & a_{1n}^* & b_1^* \\ 0 & \dots & 0 & a_{nn}^* & b_n^* \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & a_{22}^* & \dots & a_{2n}^* & b_2^* \end{array}\right]$$

So given

$$\left[\begin{array}{cccc|c} a_{11}^* & a_{12}^* & \dots & a_{1n}^* & b_1^* \\ 0 & a_{22}^* & \dots & a_{2n}^* & b_2^* \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & a_{nn}^* & b_n^* \end{array}\right],$$

we solve the equation in row n for x_n. (i.e., solve $a_{nn}^* x_n = b_n^*$ for x_n)

$$x_n = \frac{b_n^*}{a_{nn}^*}.$$

Having solved for x_n, we proceed to solve the equation in row $(n-1)$ and solve for x_{n-1}. (i.e., solve $a_{n-1,n-1}x_{n-1} + a_{n-1,n}x_n = b_{n-1}$)

$$x_{n-1} = \frac{b_{n-1} - a_{n-1,n}x_n}{a_{n-1,n-1}}$$

Inductively, we solve for x_i as follows:

$$x_i = \frac{b_i - a_{i,i+1}x_{i+1} - \ldots - a_{i,n-1}x_{n-1} - a_{i,n}x_n}{a_{i,i}}.$$

This process is called **"backward substitution."**