3.1: 1, 3, 5, 9, 10, 12, 14, 18

1) We want to solve $-\frac{d}{dx}\left[c(x)\frac{du}{dx}\right] = f(x)$ with $c(x) = c = \textit{constant}$ and $f(x) = 1 - x$ for different boundary conditions to get $w(x)$ and $u(x)$.

$$\frac{dw}{dx} = x - 1$$
$$\int_x^1 \frac{dw}{dx}dx = \int_x^1 (x-1)dx$$
$$w(1) - w(x) = \left[\frac{1}{2}x^2 - x\right]_x^1$$
$$w(x) = w(1) - \frac{1}{2}(x-1)^2$$
$$c\int_0^x \frac{du}{dx}dx = \int_0^x \left[w(1) - \frac{1}{2}(x-1)^2\right]dx$$
$$u(x) - u(0) = \frac{w(1)}{c}x - \frac{1}{6c}\left[(x-1)^3 + 1\right]$$

(a) Boundary conditions: u(0) = 0 and w(1) = 0.

$$u(x) - \cancel{u(0)} = \frac{\cancel{w(1)}}{c}x - \frac{1}{6c}\left[(x-1)^3 + 1\right] \qquad w(x) = \cancel{w(1)} - \frac{1}{2}(x-1)^2$$
$$\boxed{u(x) = \frac{1}{6c}\left[(1-x)^3 - 1\right]} \qquad \boxed{w(x) = -\frac{1}{2}(1-x)^2}$$

(b) Boundary conditions: u(0) = 0 and u(1) = 0.

$$u(x) - \cancel{u(0)} = \frac{w(1)}{c}x - \frac{1}{6c}\left[(x-1)^3 + 1\right]$$
$$u(1) = 0 = \frac{w(1)}{c} - \frac{1}{6c} \longrightarrow w(1) = \frac{1}{6} \qquad w(x) = w(1) - \frac{1}{2}(x-1)^2$$
$$\boxed{u(x) = \frac{1}{6c}\left[(1-x)^3 - (1-x)\right]} \qquad \boxed{w(x) = \frac{1}{6} - \frac{1}{2}(1-x)^2}$$

3) Looking at the case of a bar free at both ends, $-\frac{dw}{dx} = f(x)$ with $w(0) = w(1) = 0$, we want to know under which conditions a solution exists. Integrating both sides of the differential equation from 0 to 1 gives

$$-\int_0^1 \frac{dw}{dx}dx = \int_0^1 f(x)dx$$
$$-\cancel{w(1)} + \cancel{w(0)} = \int_0^1 f(x)dx$$

So, for there to be a solution to this problem, $f(x)$ must satisfy the condition $\int_0^1 f(x)dx = 0$. Physically, this means that the net force on the bar must be zero.

5) Again we have $-\frac{dw}{dx} = f(x)$ and $c(x)\frac{du}{dx} = w$, with boundary conditions $u(0) = 0$ and $w(1) = 0$. This time, $f(x) = f = \textit{constant}$ and $c(x)$ jumps from $c = 1$ for $x \le \frac{1}{2}$ to $c = 2$ for $x > \frac{1}{2}$. So, $c(x) = 1 + S(x - \frac{1}{2})$.

$$-\int_x^1 \frac{dw}{dx}dx = \int_x^1 f dx$$
$$-\cancel{w(1)} + w(x) = (1-x)f$$
$$\boxed{w(x) = (1-x)f}$$

$$\frac{du}{dx} = \frac{w(x)}{c(x)} = f\frac{1-x}{1+S(x-\frac{1}{2})}$$

$$\int_0^x \frac{du}{dx} = \begin{cases} f\int_0^x (1-x)dx & \text{for } 0 \le x \le \frac{1}{2} \\ f\int_0^{\frac{1}{2}}(1-x)dx + \frac{f}{2}\int_{\frac{1}{2}}^x (1-x)dx & \text{for } \frac{1}{2} < x \le 1 \end{cases}$$

$$u(x) - \cancel{u(0)} = \begin{cases} f\left[x - \frac{1}{2}x^2\right] & \text{for } 0 \le x \le \frac{1}{2} \\ \frac{3f}{8} + \frac{f}{2}\left[x - \frac{1}{2}x^2 - \frac{3}{8}\right] & \text{for } \frac{1}{2} < x \le 1 \end{cases}$$

$$\boxed{u(x) = \begin{cases} f\left[x - \frac{1}{2}x^2\right] & \text{for } 0 \le x \le \frac{1}{2} \\ f\left[\frac{3}{16} + \frac{1}{2}x - \frac{1}{4}x^2\right] & \text{for } \frac{1}{2} < x \le 1 \end{cases}}$$

9) The solution to $-cu'' + u' = 1$ is $u(x) = d_1 + d_2\exp(\frac{x}{c}) + x$. With the boundary conditions $u(0) = u(1) = 0$, d_1 and d_2 can be found as follows.

$$u(0) = d_1 + d_2 = 0 \longrightarrow \quad d_2 = -d_1$$

$$u(1) = d_1 + d_2\exp(\frac{1}{c}) = -1 \longrightarrow \quad d_1\left(1 - \exp(\frac{1}{c})\right) = -1 \longrightarrow \quad \boxed{-d_1 = d_2 = \frac{1}{1-\exp(\frac{1}{c})}}$$

Now, we can find the limit of d_1, d_2, and $u(x)$ as $c \to 0$ from the positive side.

$$\lim_{c\to 0^+} -d_1 = \lim_{c\to 0^+} d_2 = \lim_{c\to 0^+} \frac{1}{1-\exp(\frac{1}{c})} = \lim_{c\to 0^+} \frac{\exp(-\frac{1}{c})}{\exp(-\frac{1}{c}) - 1} = \frac{0}{0-1} = 0$$

$$\lim_{c\to 0^+} u(x) = x + \cancel{\lim_{c\to 0^+} d_1} + \lim_{c\to 0^+} d_2\exp(\frac{x}{c}) = x + \lim_{c\to 0^+} \frac{\exp(-\frac{(1-x)}{c})}{\exp(-\frac{1}{c}) - 1} = \begin{cases} 1 + \frac{1}{0-1} & \text{for } x = 1 \\ x + \frac{0}{0-1} & \text{for } 0 \le x \ll 1 \end{cases}$$

$$\lim_{c\to 0^+} u(x) = \begin{cases} 0 & \text{for } x = 1 \\ x & \text{for } 0 \le x \ll 1 \end{cases}$$

When x is away from $x = 1$ the limit of $u(x)$ as $c \to 0^+$ clearly goes to x. However, near $x = 1$ the limit rapidly diverges from x and heads to 0. And if $x \to 1$ much faster than $c \to 0$, then the limit of $u(x)$ will go to 0. The boundary condition $u(0) = 0$ is always satisfied but the end $x = 1$ has a boundary layer.

10) We want to solve $-u'' = 2$ with $u(0) = u(1) = 0$ using finite elements with three hat functions ($\phi_i(x)$) and $h = \frac{1}{4}$. So, $c(x) = 1$ and $f(x) = 2$, and we need to calculate the components of the matrix K and vector F in the equation $KU = F$ where $U = (U_1, U_2, U_3)$ and

$$K_{ij} = \int_0^1 c(x)\frac{d\phi_j}{dx}\frac{dv_i}{dx} \qquad F_i = \int_0^1 f(x)\phi_i(x)dx \qquad U(x) = U_1\phi_1(x) + U_2\phi_2(x) + U_3\phi_3(x)$$

The test functions $v_i(x)$ are taken to be the trial functions $\phi_i(x)$, or $v_i(x) = \phi_i(x)$. This makes the K matrix symmetric. The components of K and F are given below.

$$F_1 = F_2 = F_3 = 2h = \frac{1}{2} \qquad K_{11} = K_{22} = K_{33} = 2h\left(\frac{1}{h^2}\right) = 8 \qquad K_{13} = K_{31} = 0$$

$$K_{12} = K_{21} = h\left(-\frac{1}{h^2}\right) = -4 \qquad K_{23} = K_{32} = h\left(-\frac{1}{h^2}\right) = -4$$

Now, we can solve the matrix equation $KU = F$ to get the vector U and the finite element solution $U(x)$.

$$KU = F \longrightarrow \begin{bmatrix} 8 & -4 & 0 \\ -4 & 8 & -4 \\ 0 & -4 & 8 \end{bmatrix}\begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \longrightarrow \begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix} = \frac{1}{16}\begin{bmatrix} 3 \\ 4 \\ 3 \end{bmatrix}$$

Finally, we can compare to the finite element solution to the exact solution $u(x) = x - x^2$. Clearly $U(x)$ cannot equal $u(x)$ since $u(x)$ is quadratic and $U(x)$ is piecewise linear. However, we can compare the value at the mesh points: $x = \frac{1}{4}$, $x = \frac{1}{2}$, and $x = \frac{3}{4}$. It is shown below that the values of U(x) and u(x) are identical at the meshpoints.

$$u(h) = u\left(\frac{1}{4}\right) = \frac{3}{16} = U_1 \ \checkmark \qquad u(2h) = u\left(\frac{1}{2}\right) = \frac{4}{16} = U_2 \ \checkmark \qquad u(3h) = u\left(\frac{3}{4}\right) = \frac{3}{16} = U_3 \ \checkmark$$

12) The mass matrix M is defined as $M_{ij} = \int V_i V_j dx$. We would like to know the mass matrix for the three hat functions. So $V_i(x)$ would be a hat function centered about $x = hi$, where h is the spacing between mesh points. It is clear that M is symmetric, so $M_{ij} = M_{ji}$. Also, the integral of the product of two hat functions is non-zero only if the two hat functions are the same or are adjacent; in other words, $M_{ij} = 0$ if $|i - j| > 1$. So, M is a tri-diagonal, symmetric matrix when hat functions are used. Furthermore, since the integral of the square of a hat function is identical regardless of the which hat function is used, the values along the main diagonal are all equal to the same value (calculated to be $\frac{2h}{3}$). Also, since the integral of the product of two adjacent hat functions is identical regardless of which two adjacent pairs of hat functions is selected, the values along the diagonal adjacent to the main diagonal are all equal to the same value (calculated to be $\frac{h}{6}$). For the specific case of three hat functions dividing the interval 0 to 1 with $h = \frac{1}{4}$, we get the following 3×3 matrix for M:

$$\boxed{M = \frac{1}{24}\begin{bmatrix} 4 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 4 \end{bmatrix}}$$

14) We can use Simpson's $\frac{1}{6}, \frac{4}{6}, \frac{1}{6}$ rule to integrate and find the area underneath the bubble function $\phi_5(x)$.

$$\int_h^{2h} \phi_5(x)dx = \frac{h}{6}\phi_5(h) + \frac{4h}{6}\phi_5(\tfrac{3h}{2}) + \frac{h}{6}\phi_5(2h) = \frac{h}{6}(0) + \frac{4h}{6}(1) + \frac{h}{6}(0) = \boxed{\frac{2h}{3}}$$

18) To find the finite element solution to a fixed-free hanging bar with $c(x) = 1$ (boundary conditions $u(0) = 0$ and $u'(1) = 0$), we must add an additional half-hat function, $\phi_{N+1}(x)$ at the end in addition to the N interior hat functions $\phi_i(x)$. The spacing between meshpoints is $h = \frac{1}{N+1}$ and the test functions are taken to be equal to the trial functions ($V_i(x) = \phi_i(x)$).

(a) The stiffness matrix K would now be $(N+1) \times (N+1)$ instead of $N \times N$ as in the fixed-fixed case. The extra row and column mostly have zero entries except near the lower right corner. The entry in row N, column $(N+1)$ (which is identical to the entry in row $(N+1)$, column N because K is symmetric) would be equal to $-\frac{1}{h}$. The entry in row $(N+1)$, column $(N+1)$ would be equal to $\frac{1}{h}$. So multiplying the column vector U with the $(N+1)$ row of K gives the expression

$$\frac{U_{N+1} - U_N}{h}$$

which is a first difference of U; specifically, it is a backward difference of U at the point $x = 1$, which is analogous to $u'(1)$. This last row of the stiffness matrix is representing the boundary condition $u'(1) = 0$.

(b) With a constant load $f(x) = f_0$, the new last component of the column vector F would be $F_{N+1} = \frac{1}{2}hf_0$. The 3×3 matrix K and 3×1 column vector F from problem 10 can be generalized for the $N \times N$ case. But in addition to that, we must add the extra row and column to get the larger $(N+1) \times (N+1)$ stiffness matrix K and the $(N+1) \times 1$ column vector F for this fixed-free case. The form for K and F (for a fixed-free hanging bar with $c(x) = 1$ and $f(x) = f_0$) are given by

$$K = \frac{1}{h}\begin{bmatrix} 2 & -1 & & & & \\ & -1 & 2 & -1 & & \\ & & & \ddots & & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 1 \end{bmatrix} \qquad F = hf_0\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ \frac{1}{2} \end{bmatrix}$$

Now, we can solve $KU = F$ to get the column vector U and the finite element solution $U(x) = \Sigma_{i=1}^{(N+1)} U_i \phi_i(x)$, and then compare it with the true solution of $u(x) = f_0(x - \frac{1}{2}x^2)$. Particularly, we would like to compare the values evaluated at the mesh points: for the finite element solution this is simply given by U_i; for the true solution it is given by $u(hi) = \frac{1}{2}h^2 f_0(\frac{2}{h} - i)(i) = \frac{1}{2}h^2 f_0[2(N+1) - i](i)$.

$$U = K^{-1}F \longrightarrow \begin{bmatrix} U_1 \\ \vdots \\ U_N \\ U_{N+1} \end{bmatrix} = h^2 f_0 \begin{bmatrix} 1 & 1 & \dots & 1 & 1 \\ 1 & 2 & \dots & 2 & 2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 2 & \dots & N & N \\ 1 & 2 & \dots & N & (N+1) \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ \frac{1}{2} \end{bmatrix}$$

Through some careful counting we can see that for $i = 1, ..., (N+1)$:

$$\begin{aligned} U_i &= h^2 f_0 \left[(N+1-i)i - \frac{1}{2}i + \Sigma_{j=1}^{i} j \right] \\ &= h^2 f_0 \left[(N+1)i - i^2 - \frac{1}{2}i + \frac{(i)(i+1)}{2} \right] \\ &= \frac{1}{2}h^2 f_0 \left[2(N+1)i - 2i^2 - i + i^2 + i \right] \\ U_i &= \frac{1}{2}h^2 f_0 \left[2(N+1) - i \right](i) \end{aligned}$$

This equation is identical to the equation for $u(hi)$, so $u(hi) = U_i$.

3.2: 1, 3, 5, 6

1) We want to solve the equation for a cantilevered beam with a unit force at the midpoint, $u''''(x) = \delta(x - \frac{1}{2})$ with boundary conditions $u(0) = u'(0) = 0$ and $M(1) = M'(1) = 0$. The differential equation can be split into $M''(x) = \delta(x - \frac{1}{2})$ and $u''(x) = M(x)$, and then solved as follows.

$$\begin{aligned} \int_x^1 M''(x)dx &= \int_x^1 \delta(x - \frac{1}{2})dx \\ \cancel{M'(1)} - M'(x) &= 1 - S(x - \frac{1}{2}) \\ \int_x^1 M'(x)dx &= \int_x^1 \left(S(x - \frac{1}{2} - 1 \right) dx \\ \cancel{M(1)} - M(x) &= (\frac{1}{2} - 1) - (R(x - \frac{1}{2}) - x) \\ \int_0^x u''(x)dx &= \int_0^x \left[R(x - \frac{1}{2}) + x + \frac{1}{2} \right] dx \\ u'(x) - \cancel{u'(0)} &= Q(x - \frac{1}{2}) + \frac{1}{2}x^2 + \frac{1}{2}x \\ \int_0^x u'(x)dx &= \int_0^x \left[Q(x - \frac{1}{2})dx + \frac{1}{2}(x^2 + x) \right] dx \\ u(x) - \cancel{u(0)} &= C(x - \frac{1}{2}) + \frac{1}{2}\left(\frac{1}{3}x^3 + \frac{1}{2}x^2 \right) \end{aligned}$$

$$\boxed{u(x) = \begin{cases} \frac{1}{6}(2x^3 + 3x^2) & \text{for } 0 \le x \le \frac{1}{2} \\ \frac{1}{6}(2x^3 + 3x^2) + (x - \frac{1}{2})^3 & \text{for } \frac{1}{2} < x \le 1 \end{cases}}$$

3) We want to solve $u''''(x) = \delta(x)$ with boundary conditions $u(-1) = u'(-1) = 0$ and $u(1) = u'(1) = 0$. The general solution to $u''''(x) = \delta(x)$ is given by

$$u(x) = \begin{cases} A + Bx + Cx^2 + Dx^3 - \frac{1}{6}x^3 & \text{for } x \leq 0 \\ A + Bx + Cx^2 + Dx^3 & \text{for } x \geq 0 \end{cases}$$

We can use the boundary conditions to get four equations that can be used to solve for the four unknowns A, B, C, and D.

$$u(-1) = A - B + C - D + \frac{1}{6} = 0 \qquad u(1) = A + B + C + D = 0$$
$$u'(-1) = B - 2C + 3D - \frac{1}{2} = 0 \qquad u'(1) = B + 2C + 3D = 0$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 0 & 1 & 2 & 3 \\ 0 & 1 & -2 & 3 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = \begin{bmatrix} 0 \\ -\frac{1}{6} \\ 0 \\ \frac{1}{2} \end{bmatrix} \longrightarrow \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = \begin{bmatrix} \frac{1}{24} \\ 0 \\ -\frac{1}{8} \\ \frac{1}{12} \end{bmatrix}$$

$$\boxed{u(x) = \begin{cases} \frac{1}{24} - \frac{1}{8}x^2 - \frac{1}{12}x^3 & \text{for } x \leq 0 \\ \frac{1}{24} - \frac{1}{8}x^2 + \frac{1}{12}x^3 & \text{for } x \geq 0 \end{cases}}$$

5) Which of the 8 cubic finite elements, $\phi_0^d(x), \phi_0^s(x), ..., \phi_3^d(x), \phi_3^s(x)$ based at the meshpoints $x = 0, \frac{1}{3}, \frac{2}{3}, 1$, are dropped because of the essential boundary conditions below for $-u'' = f$?

(a) Fixed-fixed: $u(0) = u(1) = 0$.
$\phi_0^d(x)$ and $\phi_3^d(x)$ would be dropped.

(b) Fixed-free: $u(0) = u'(1) = 0$.
Only $u(0) = 0$ is an essential boundary condition. Thus, $\phi_0^d(x)$ would be dropped.

6) Which of the 8 cubic finite elements, $\phi_0^d(x), \phi_0^s(x), ..., \phi_3^d(x), \phi_3^s(x)$ based at the meshpoints $x = 0, \frac{1}{3}, \frac{2}{3}, 1$, are dropped because of the essential boundary conditions below for $u'''' = f$?

(a) Built-in beam: $u = u' = 0$ at both ends.
$\phi_0^d(x)$, $\phi_0^s(x)$, $\phi_3^d(x)$, and $\phi_3^s(x)$ would be dropped.

(b) Simply supported beam: $u = u'' = 0$ at both ends.
Only $u(0) = u(1) = 0$ are the essential boundary conditions. Thus, $\phi_0^d(x)$ and $\phi_3^d(x)$ would be dropped.

(c) Cantilevered beam: $u(0) = u'(0) = u''(1) = u'''(1) = 0$.
Only $u(0) = u'(0) = 0$ are the essential boundary conditions. $\phi_0^d(x)$ and $\phi_0^s(x)$ would be dropped.

MATLAB Assignment

We want to convert between a geocentric rectangular coordinate system and a geodetic coordinate system. This conversion is useful in GPS applications. The conversion from geodetic coordinates (latitude ϕ, longitude λ, altitude h) to geocentric rectangular coordinates (X, Y, Z) is straightforward and is given by

$$X = (N+h)\cos\phi\cos\lambda \tag{1}$$
$$Y = (N+h)\cos\phi\sin\lambda \tag{2}$$
$$Z = \left[(1-f)^2 N + h\right]\sin\phi \tag{3}$$

where
$$N = a\left[1 - f(2-f)\sin^2\phi\right]^{-\frac{1}{2}} \tag{4}$$

and $a = 6378388\ m$ is the length of the semi-major axis of the Earth and $f = \frac{1}{297}$ is the polar flattening of the Earth. The conversion from geocentric rectangular coordinates to geodetic coordinates is more difficult because it requires solving a system of non-linear equations.

The number of equations to solve can be reduced from three to two by solving for λ analytically. By substituting equations 1 and 2 into the quotient $\frac{Y}{X}$ we get

$$\tan\lambda = \frac{Y}{X} \tag{5}$$

By squaring both sides of equations 1 and 2, adding the two resulting equations, and finally taking the square root of both sides of the summed equation, we get

$$\sqrt{X^2+Y^2} = (N+h)\cos\phi\sqrt{\cos^2\lambda + \sin^2\lambda} = (N+h)\cos\phi$$

The above equation is used to define the function $g_1(\phi, h)$, and equation 3 is used to define the function $g_2(\phi, h)$:

$$g_1(\phi,h) \equiv (N+h)\cos\phi - \sqrt{X^2+Y^2} = 0 \tag{6}$$
$$g_2(\phi,h) \equiv \left[(1-f)^2 N + h\right]\sin\phi - Z = 0 \tag{7}$$

where N is given in equation 4.

Now, we need to calculate the Jacobian,

$$J = \begin{bmatrix} \frac{\partial g_1}{\partial\phi} & \frac{\partial g_1}{\partial h} \\ \frac{\partial g_2}{\partial\phi} & \frac{\partial g_2}{\partial h} \end{bmatrix}$$

The partial derivatives with respect to altitude h are easy to calculate. They are simply $\frac{\partial g_1}{\partial h} = \cos\phi$ and $\frac{\partial g_2}{\partial h} = \sin\phi$. To simplify calculating the partial derivatives with respect to ϕ, we will first calculate $\frac{\partial N}{\partial\phi}$.

$$\frac{\partial N}{\partial\phi} = -\frac{1}{2}a\left[1 - f(2-f)\sin^2\phi\right]^{-\frac{3}{2}}\left(-2f(2-f)\sin\phi\cos\phi\right) = \frac{N^3}{a^2}f(2-f)\sin\phi\cos\phi$$

$$\begin{aligned}
\frac{\partial g_1}{\partial\phi} &= \frac{\partial N}{\partial\phi}\cos\phi - (N+h)\sin\phi = \frac{N^3}{a^2}f(2-f)\sin\phi\cos^2\phi - N\sin\phi - h\sin\phi \\
&= \frac{N^3}{a^2}\sin\phi\left[f(2-f)\cos^2\phi - \frac{a^2}{N^2}\right] - h\sin\phi \\
&= \frac{N^3}{a^2}\sin\phi\left[f(2-f)\cos^2\phi - 1 + f(2-f)\sin^2\phi\right] - h\sin\phi \\
&= \frac{N^3}{a^2}\sin\phi\left[-1 + 2f - f^2\right] - h\sin\phi = -\left[\frac{N^3}{a^2}(1-f)^2 + h\right]\sin\phi
\end{aligned}$$

$$\begin{aligned}
\frac{\partial g_2}{\partial \phi} &= (1-f)^2 \frac{\partial N}{\partial \phi} \sin\phi + \left[(1-f)^2 N + h\right] \cos\phi \\
&= (1-f)^2 \left[\frac{N^3}{a^2} f(2-f) \sin^2\phi \cos\phi\right] + (1-f)^2 N \cos\phi + h\cos\phi \\
&= (1-f)^2 \frac{N^3}{a^2} \cos\phi \left[f(2-f)\sin^2\phi + \frac{a^2}{N^2}\right] + h\cos\phi \\
&= (1-f)^2 \frac{N^3}{a^2} \cos\phi \left[f(2-f)\sin^2\phi + 1 - f(2-f)\sin^2\phi\right] + h\cos\phi \\
&= (1-f)^2 \frac{N^3}{a^2} \cos\phi + h\cos\phi = \left[\frac{N^3}{a^2}(1-f)^2 + h\right] \cos\phi
\end{aligned}$$

If we define $D \equiv \frac{N^3}{a^2}(1-f)^2 + h$, then the Jacobian can be simply written as

$$J = \begin{bmatrix} -D\sin\phi & \cos\phi \\ D\cos\phi & \sin\phi \end{bmatrix}$$

and the inverse of the Jacobian is

$$\begin{aligned}
J^{-1} &= \frac{1}{\det(J)} \begin{bmatrix} \sin\phi & -\cos\phi \\ -D\cos\phi & -D\sin\phi \end{bmatrix} = \frac{1}{-D\sin^2\phi - D\cos^2\phi} \begin{bmatrix} \sin\phi & -\cos\phi \\ -D\cos\phi & -D\sin\phi \end{bmatrix} \\
&= \frac{1}{-D} \begin{bmatrix} \sin\phi & -\cos\phi \\ -D\cos\phi & -D\sin\phi \end{bmatrix} = \begin{bmatrix} -\frac{1}{D}\sin\phi & \frac{1}{D}\cos\phi \\ \cos\phi & \sin\phi \end{bmatrix}
\end{aligned}$$

Now, we can find the solution vector $u = (\phi, h)$ using Newton's Method. We start with an initial guess vector u_{init} and set the current vector u_{cur} equal to it. For each iteration in the loop, we compute a new vector u_{new} through the computation

$$u_{\text{new}} = u_{\text{cur}} - J^{-1} g \xrightarrow{or} \begin{bmatrix} \phi_{\text{new}} \\ h_{\text{new}} \end{bmatrix} = \begin{bmatrix} \phi_{\text{cur}} \\ h_{\text{cur}} \end{bmatrix} - \begin{bmatrix} -\frac{1}{D}\sin\phi_{\text{cur}} & \frac{1}{D}\cos\phi_{\text{cur}} \\ \cos\phi_{\text{cur}} & \sin\phi_{\text{cur}} \end{bmatrix} \begin{bmatrix} g_1(\phi_{\text{cur}}, h_{\text{cur}}) \\ g_2(\phi_{\text{cur}}, h_{\text{cur}}) \end{bmatrix} \tag{8}$$

where

$$D = \frac{N^3}{a^2}(1-f)^2 + h \tag{9}$$

and the functions g_1 and g_2 are defined in equations 6 and 7 respectively, except the functions would of course not equal zero when evaluated at the current u_{cur} unless u_{cur} happened to exactly be the true solution. Every time the loop is repeated, u_{new} of the prior loop becomes u_{cur} of the current loop. The new calculated vector u_{new} should be closer to the actual solution. Eventually, u_{new} should be close enough to the actual solution, within some tolerance, that we can break out of the loop and treat u_{new} as the solution. The MATLAB function that does this conversion essentially uses equations 1 to 4 to convert the current estimate for the geodetic coordinates into geocentric rectangular coordinates (X_{cur}, Y_{cur}, Z_{cur}), and then it calculates the Euclidean distance between this point and the point (X, Y, Z) specified by the user. If this distance is smaller than some threshold distance, which can be specified by the user through an optional argument in the MATLAB function, it will break out of the loop. Newton's method requires an initial guess to start from; however, testing showed that regardless of where the initial guess was, the unique correct solution was quickly obtained in a few iterations. Therefore, the MATLAB function makes specifying the initial guess of the latitude and altitude optional, and by default it sets both to zero (at the equator with zero altitude). Finally, the longitude is explicitly calculated from X and Y alone, using equation 5.

The code for the MATLAB function, `gps`, that converts from geocentric rectangular to geodetic coordinates is displayed in Listing 1.

We can look at an example where we wish to convert the geocentric rectangular coordinates (X, Y, Z) to geodetic coordinates latitude, longitude, and altitude. X, Y, and Z are specified as follows:

$$\begin{aligned} X &= \quad 3427056.327\ m \\ Y &= \quad 601199.931\ m \\ Z &= \quad 5327877.892\ m \end{aligned}$$

Then the following MATLAB code can be used to call the function and get the converted coordinates:

```
[phi, lambda, h] = gps( 3427056.327, 601199.931, 5327877.892 )
```

The variable `h` has the altitude (h) in meters, the variable `phi` has the latitude (ϕ) in degrees, and the variable `lambda` has the longitude (λ) in degrees. The values for these three variables, for this example, are

$$\begin{aligned} \phi &\approx 57.03^\circ \\ \lambda &\approx 9.95^\circ \\ h &\approx 56.95\ m \end{aligned}$$

We can verify the results of the function by plugging these values into equations 1 to 4 and getting the original specified values for X, Y, and Z.

Listing 1: MATLAB function that converts from geocentral rectangular to geodetic coordinates.

```
function [ lat, long, alt ] = gps( X, Y, Z, varargin )
%GPS    Converts from geocentric rectangular to geodetic coordinates.
%   [lat, long, alt] = gps (X, Y, Z, init_lat = 0, init_alt = 0, ...
%                           tol = 1e-6, max_iter = 100)
%   converts from geocentric rectangular coordinates (X, Y, Z) in meters to
%   geodetic coordinates: latitude (lat) and longitude (long) in degrees
%   and altitude (alt) in meters. The conversion is done by solving a
%   system of non-linear equations using Newton's method, so an initial
%   guess for the latitude (init_lat) in degrees and the altitude
%   (init_alt) in meters is necessary. By default the initial latitude is
%   set to 0 degrees (equator) and the initial altitude is set to 0 m.
%
%   The tolerance (tol), which specifies the maximum acceptible distance in
%   meters between the specified (X, Y, Z) point and the estimated point
%   given by the calculated latitude, longitude, altitude, is set to 1e-6 m
%   by default. The maximum number of iterations (max_iter) before aborting
%   Newton's method and printing an error message is set to 100 by default.

% Constants for Earth
a = 6378388; % [m] length of semi-major axis of Earth
f = 1/297;   % polar flattening

% Set default values for tolerance and max_iter if not specified.
numvarargs = length(varargin);
if (numvarargs > 4)
    error('Requires at most 4 optional inputs');
end
optargs = {0, 0, 1e-6, 100}; % Set defaults
newVals = cellfun(@(x) ~isempty(x), varargin);
optargs(newVals) = varargin(newVals);
[init_lat, init_alt, tol, max_iter] = optargs{:};

```

```
33 % Newton's Method
34 u = [degtorad(init_lat) ; init_alt];
35 g_right = [ sqrt(X*X + Y*Y) ; Z ];
36 g_right_norm = norm(g_right); % g_right_norm = sqrt(X*X + Y*Y + Z*Z);
37 for i=[1:max_iter]
38     phi = u(1); h = u(2);
39     N = a*(1 - f*(2-f)*(sin(phi)^2))^(-1/2);
40     D = h + (1-f)^2 * N^3 / a^2;
41     g_left = [ (N+h)*cos(phi) ; (N*(1-f)^2 + h)*sin(phi) ];
42     g  = g_left - g_right;
43     u  = u - [-sin(phi)/D, cos(phi)/D ; cos(phi), sin(phi)] * g;
44
45     % Calculate current distance from desired point (error) and check if it
46     % is within the specified tolerance.
47     if (abs(norm(g_left) - g_right_norm) < tol)
48         break; % Tolerance satisfied, break out of loop.
49     end
50 end
51
52 % Check if loop aborted because of maximum iterations
53 if (i >= max_iter)
54     error('gps:maxIteration', ...
55         ['Exceeded maximum number of iterations (%d) without finding', ...
56          ' a solution within the tolerance'], max_iter);
57 end
58
59 % Set output variables
60 long = radtodeg(atan2(Y, X));
61 lat  = radtodeg(phi);
62 alt  = h;
63 end
```