1 Review of vector calculus.

The following is a brief refresher of vector calculus to the extent that we'll be using it in class.

Notation:

\[ \mathbf{r} = (r_1, r_2, r_3) = (x, y, z) \]
would typically denote a position vector. (You may think of this as a column vector; \( r^T \) would then be a row vector.)

\[ \hat{\mathbf{r}} \]
is a unit vector.

\[ \mathbf{a} \mathbf{b} = a_i b_j \]
is a vector dyad (notation as in Problem 7). In matrix representation, we would write \( \mathbf{a} \mathbf{b}^T \) showing how it can be obtained as the matrix product of a column and a row vector. It is indeed the tensor (or matrix) with \( a_i b_j \) as the \( i^j \)th element.

Let \( U(x, y, z) \) be a scalar function of position.
Let \( \mathbf{B}(x, y, z) = (B_x(x, y, z), B_y(x, y, z), B_z(x, y, z)) \) be a vector function of position. It defines a vector at every point in space, \( B_x(x, y, z) \hat{x} + B_y(x, y, z) \hat{y} + B_z(x, y, z) \hat{z} \).

The gradient operator \( \nabla \) has components \( \left( \frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right) \). It augments the order of a tensor with one: from our scalar function \( U \), it defines the vector field

\[
\nabla U = \begin{pmatrix} \frac{\partial U}{\partial x} & \frac{\partial U}{\partial y} & \frac{\partial U}{\partial z} \end{pmatrix}
\]

It makes a second-order tensor from a vector, and so on.
The **divergence** operator \( \nabla \cdot \) (the **dot product** of the gradient with the argument) reduces the order with one - from our **vector** field, it makes a **scalar** field.

\[
\nabla \cdot \mathbf{B} = \frac{\partial B_x}{\partial x} + \frac{\partial B_y}{\partial y} + \frac{\partial B_z}{\partial z} \tag{2}
\]

The divergence measures **sources** and **sinks** in the material. In continuum mechanics, the best example of what this means is: a material with invariant volume (incompressible) has a velocity field (specifying the velocity of each particle at each point in space) which is said to be **divergence-free**: \( \nabla \cdot \mathbf{u} = 0 \).

The **Laplacian** operator \( \nabla \cdot \nabla \) or \( \nabla^2 \) leaves the order of a tensor intact. For the scalar function \( U \),

\[
\nabla^2 U = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial z^2} \tag{3}
\]

The **curl** or rotation operator \( \nabla \times \) (the **cross product** of the gradient with the argument) also leaves the order intact - for our vector function \( \mathbf{B} \), the easiest representation is in **determinant** form:

\[
\nabla \times \mathbf{B} = \begin{vmatrix}
\frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\
\frac{\partial}{\partial z} & \frac{\partial}{\partial x} & \frac{\partial}{\partial y} \\
B_x & B_y & B_z
\end{vmatrix} \tag{4}
\]

which is as much as

\[
\left( \frac{\partial B_z}{\partial y} - \frac{\partial B_y}{\partial z} \right) \hat{x} + \left( \frac{\partial B_x}{\partial z} - \frac{\partial B_z}{\partial x} \right) \hat{y} + \left( \frac{\partial B_y}{\partial x} - \frac{\partial B_x}{\partial y} \right) \hat{z} \tag{5}
\]

which defines a vector field normal to \( \mathbf{B} \) and its gradient. It measures the **vorticity** of the \( \mathbf{B} \)-field (see problem 6 for a clarifying example of what this means).

2
2 Problems.

Let \( U = 2 \frac{z}{y} + 2 \frac{xy^3}{x^2} + 3xz^4 \)
Let \( B = \frac{z}{x} \hat{x} + 2y^3z \hat{y} + 2y^2z^3 \hat{z} \)

1. Calculate the gradient of \( U \).
2. Calculate the divergence of \( B \).
3. Calculate the Laplacian of \( U \).
4. Verify that the curl of the gradient of \( U \) is 0.
5. Verify that \( B \cdot (B \times \nabla U) \) is 0.
6. Let \( \mathbf{u}^e = \omega (-y \hat{x} + x \hat{y}) \) describe a velocity field in a material.
   (a) What kind of motion is this material undergoing?
   (b) Make a plot of this vector field.
   (c) Calculate \( \frac{1}{2} \nabla \times \mathbf{u}^e \). Explain what you get.
7. Let \( \mathbf{I} \) be the identity tensor/matrix. What kind of an operator is
   \[
   (\mathbf{I} - 2\hat{z}) \cdot ?
   \]  
(6)
8. (a) Calculate the (scalar) moment of inertia (around one axis) of a spherically symmetric body with constant density.
   (b) Now derive an expression for the moment of inertia for spherically symmetric bodies with a two-step variation of density. Apply this to a planet with a uniform-density mantle and a uniform core of half the total radius \( R \), and with a density that is \( f \) times the mantle density. What values of \( f \) would be required to give moments of inertia of \( 293/886 \, MR^2 \), \( 73/200 \, MR^2 \), and \( 391/1000 \, MR^2 \), corresponding to Earth, Mars and Moon, respectively?
   (c) The next step is a continuous, functional variation of density with radius. The following are data on the density variations within the Sun.
\[
\begin{array}{|c|c|}
\hline
\frac{r}{r_s} & \rho \text{ (kgm}^{-3}\text{)} \\
\hline
0 & 160,000 \\
0.04 & 141,000 \\
0.1 & 89,000 \\
0.2 & 41,000 \\
0.3 & 13,300 \\
0.4 & 3,600 \\
0.5 & 1,000 \\
0.6 & 350 \\
0.7 & 80 \\
0.8 & 18 \\
0.9 & 2 \\
0.95 & 0.4 \\
1.0 & 0 \\
\hline
\end{array}
\]

Assume a monotonous decrease of density with distance from the center. Describe \( \rho(r) \) functionally by fitting a low-order polynomial through the data. (Note: MATLAB’s function polyfit might come in handy here. Use these expressions to obtain an estimate of the moment of inertia of the Sun. A detailed estimate yields 5.7 \times 10^{46} \text{ kgm}^2. ) What fraction is this value of the moment of inertia of a uniform sphere of the same mass and radius?

The outer radius is \( r_s = 6.96 \times 10^8 \text{ m} \) and the total mass is \( M_s = 1.989 \times 10^{30} \text{ kg} \).
3 Matlab

We don’t want to require you to use MATLAB™ but in fact, for many problems, there will be a part where you can find out how fun MATLAB™ really can be. For your convenience & entertainment, see what happens if you run the following programs...

```
% Program jello.m

clear all
x1=2:2:10;
x2=2*ones(size(x1));
X=[x1; x2];
t=0:0.05:1;
om=2*pi;

for ind=1:length(X),
    r1=X(1,ind)*cos(om*t)+X(2,ind)*sin(om*t);
    r2=-om*X(1,ind)*sin(om*t)+om*X(2,ind)*cos(om*t);
    v1=-om*X(1,ind)*cos(om*t)-om*X(2,ind)*sin(om*t);
    v2=om*X(1,ind)*cos(om*t)+om*X(2,ind)*sin(om*t);
    R1(:,ind)=r1';
    R2(:,ind)=r2';
    V1(:,ind)=v1';
    V2(:,ind)=v2';
end

quiver(R1,R2,V1,V2,0.75)
axis([-10 10 -10 10])
axis('square')
grid

% Program flow.m

clear all
r1=-5:1:5;
r2=-5:1:5;
[R1,R2]=meshgrid(r1,r2);
v1=r1;
v2=-r2;
[V1,V2]=meshgrid(v1,v2);
quiver(R1,R2,V1,V2,2)
axis([-6 6 -5 5])
axis('square')
grid
```