

Appendix A: Function Specification for MATLAB Code Provided

```
function [shearStress] = magneticShear(magnetThickness,statorCurrent)
%MAGNETICSHEAR Calculate the magnetic shear stress acting on the rotor
% [shearStress] = magneticShear(magnetThickness,statorCurrent) calculates
% the magnetic shear stress (units of Pa) acting on the surface of the
% electric machine's rotor.
%
% magnetThickness: thickness of surface mounted PMs. Unit: meters
% statorCurrent: per unit value, normalized by rated current (1: rated
% current, 0: no current)
```

```
function [loss] = rotorLosses(magnetThickness, ...
                             rotorDiameter, ...
                             axialLength, ...
                             statorCurrent, ...
                             rotorSpeed)
%ROTORLOSSES Calculate total losses on rotor
% [loss] = rotorLosses(magnetThickness, rotorDiameter, axialLength,
% statorCurrent, rotorSpeedDetailed)
%
% magnetThickness: thickness of surface mounted PMs. Unit: meters
% rotorDiameter: outer diameter of magnets. Unit: meters
% axialLength: axial length of the magnets. Unit: meters
% statorCurrent: per unit value, normalized by rated current
% (1: rated current, 0: no current)
% rotorSpeed: rotational speed of the rotor. Units: rev/min.
```

```
function [loss] = statorLosses(magnetThickness, ...
                              rotorDiameter, ...
                              axialLength, ...
                              statorCurrent, ...
                              rotorSpeed)
%STATORLOSSES Calculate losses in the electric machine's stator
% statorLosses(magnetThickness, rotorDiameter, axialLength, ...
% statorCurrent, rotorSpeed)
% magnetThickness: thickness of surface mounted PMs. Unit: meters
% rotorDiameter: outer diameter of magnets. Unit: meters
% axialLength: axial length of the magnets. Unit: meters
% statorCurrent: per unit value, normalized by rated current
% (1: rated current, 0: no current)
% rotorSpeed: rotational speed of the rotor. Units: rev/min.
```

```

function [params] = ambParameters(rotorDiameter,forceRating)
%AMBPARAMETERS Calculate the parameters of a desired magnetic bearing
% [params] = ambParameters(rotorDiameter,forceRating)
%     rotorDiameter: diameter of shaft. Units: meters
%     forceRating: force rating needed for AMB. Units: newtons
% params - structure with the following fields:
%     - stiffnessConstant: units of N/m
%     - forceConstant: units of N/A
%     - biasCurrent: units of A
%     - ratedControlCurrent: units of A
%     - coilInductance: units of H
%     - coilResistance: units of Ohms
%     - axialLength: units of meters

```

```

function [power] = baselineStorageCycle(time)
%BASELINESTORAGECYCLE Calculate the power demand for the baseline cycle
% [power] = baselineStorageCycle(time) calculates
% the grid power demand (units of W) for the 15-minute baseline
% frequency regulation cycle. Positive power values indicate a power flow
% from the storage device to the power grid.
%
% This function is vectorized and accepts time as a scalar or an array.
%
% time: time since the start of the cycle. Unit: seconds
%       (Must be between 0 and 900 seconds)

```

```

function [power_W] = team_1_cycle(time_s)
%TEAM_1_CYCLE Calculate the power demand for this team's unique cycle
% [power_W] = team_1_cycle_cycle(time_s) calculates
% the grid power demand (units of W) for this team's unique
% energy storage cycle.
%
% This function is vectorized and accepts time_s as a scalar or an array.
%
% Input Arguments:
%     time_s: time since the start of the cycle. Unit: seconds
%             (Must be between 0 and 21600.0 seconds)
%
% Return Value:
%     power_W: power demanded by the grid. Unit: Watts
%
% This cycle has a total duration of 6.0 hours (21600 seconds).
%
% This is an auto-generated wrapper function.

```

.
. .
. .
. .

```
function [power_W] = team_26_cycle(time_s)
%TEAM_26_CYCLE Calculate the power demand for this team's unique cycle
% [power_W] = team_26_cycle_cycle(time_s) calculates
% the grid power demand (units of W) for this team's unique
% energy storage cycle.
%
% This function is vectorized and accepts time_s as a scalar or an array.
%
% Input Arguments:
%   time_s: time since the start of the cycle. Unit: seconds
%           (Must be between 0 and 21600.0 seconds)
%
% Return Value:
%   power_W: power demanded by the grid. Unit: Watts
%
% This cycle has a total duration of 6.0 hours (21600 seconds).
%
% This is an auto-generated wrapper function.
```