Contents

UT Power Plant Gas Turbine Analysis (Phase 1)

assumptions: RH0 = 0%, fuel = natural gas, entire turbine is adiabatic

sb => s bar hb => h bar

```
function [PNET, mdotin, mdotout, nTH, Tturb, Teng, SFC, HR, hMain] = phase1_calcs(nT1, nT2, Tin, mdotf, Vdot1b, RPM, TSplot)
```

Set Up

```
    % general constants
    Rbar = 8.314; % kJ/kmol*K
    M = 28.02*.79 + 31.999*.21; % kg/kmol

    % conversion values
    psi2kPa = 6.895;
    R2K = 5/9;
    ft2m = 1/3.281;
    inH2O2kPa = 1/4.019;
    lb2kg = 1/2.205;
    BTUlb2kJkg = 2.326;
    hr2s = 3600;
```

Input Operating Parameters

```
    % ambient air conditions
    p1 = 14.417 * psi2kPa; % kPa
    T1 = (Tin + 459.67) * R2K; % K

    % altitude (not used in phase 1)
    ALT = 530 * ft2m; % m

    % inlet air mass flow rate
    v1 = (Rbar/M)*T1/p1;
    Vdot1 = Vdot1b * RPM/9784;
    mdot1 = Vdot1/v1; % kg/s

    % inlet pressure loss
    delPin = 4 * inH2O2kPa; % kPa

    % exhaust pressure loss
    delPex = 10 * inH2O2kPa ; % kPa

    % lower heating value
    LHV = 19000 * BTUlb2kJkg; % kJ/kg

    % fuel mass flow rate
    mdotf = mdotf * lb2kg * 1/hr2s; % kg/s
```

Input Design Parameters

```
% compressor pressure ratios
rLPC = 6;
rHPC = 4;

% bypass air mass flow percent
mBP = 0;

% HP compressor parasitic bleed air mass flow percent
mPB = 0;

% compressor efficiencies (percent)
nLPC = .82;
nHPC = .84;

% turbine efficiencies (percent)
nHPT = nT1;
nLPT = nT2;

% generator efficiency (percent)
nGEN = .977;
```

Analysis

```
% state 1
sb1 = sbarcalc(T1);

% state 1 -> 2: guide vanes
p2 = p1 - delPin;
T2 = T1;
hb2 = hbarcalc(T2);
h2 = hb2 / M;

% state 2 -> 25: LP compressor
p25 = p2 * rLPC;
sb2 = sbarcalc(T2);
sb25s = sb2 + Rbar*log(p25/p2);
T25s = TcalcS(sb25s);
hb25s = hbarcalc(T25s);
h25s = hb25s / M;
h25 = h2 - (h2 - h25s)/nLPC;
hb25 = h25 * M;
T25 = TcalcH(hb25);
WdotC1 = mdot1*(h25-h2);

% state 25 -> 3: HP compressor
p3 = p25 * rHPC;
sb25 = sbarcalc(T25);
sb3s = sb25 + Rbar*log(p3/p25);
T3s = TcalcS(sb3s);
hb3s = hbarcalc(T3s);
h3s = hb3s / M;
h3 = h25 - (h25 - h3s)/nHPC;
hb3 = h3 * M;
T3 = TcalcH(hb3);
sb3 = sbarcalc(T3);
WdotC2 = mdot1*(h3-h25);

% state 3 -> 4: combustor
mdot4 = mdot1 + mdotf;
Qdot = mdotf * LHV;
h4 = (Qdot + mdot1*h3)/mdot4;
hb4 = h4 * M;
T4 = TcalcH(hb4);
```

```
    sb4 = sbarcalc(T4);
    p4 = p3;

    % state 4 -> 48: HP turbine
    WdotT1 = WdotC1 + WdotC2;
    h48 = h4 - (WdotT1/mdot4);
    hb48 = h48 * M;
    T48 = TcalcH(hb48);
    sb48 = sbarcalc(T48);
    wT1 = WdotT1/mdot4;
    h48s = h4 - wT1/nHPT;
    hb48s = h48s * M;
    T48s = TcalcH(hb48s);
    sb48s = sbarcalc(T48s);
    p48 = p4*exp((sb48s-sb4)/Rbar);

    % state 48 -> 5: LP turbine
    p5 = p1 + delPex;
    sb5s = sb48 + Rbar*log(p5/p48);
    T5s = TcalcS(sb5s);
    hb5s = hbarcalc(T5s);
    h5s = hb5s / M;
    h5 = h48 - nLPT*(h48-h5s);
    hb5 = h5 * M;
    T5 = TcalcH(hb5);
    sb5 = sbarcalc(T5);
    WdotT2 = mdot4*(h48 - h5);
    WdotELEC = WdotT2*nGEN;

    % state 6: after nozzle
    p6 = p1;

    T6 = T5;
    sb6 = sbarcalc(T6);
```

Output Performance Parameters

```
    Tturb = T4 * 1/R2K - 459.67;
    Teng = T6 * 1/R2K - 459.67;
    PNET = WdotELEC / 1000;
    mdotin = mdot1 * 1/lb2kg * hr2s;
    mdotout = mdot4 * 1/lb2kg * hr2s;
    nTH = WdotELEC/Qdot;
    SFC = mdotf/WdotELEC * 1/lb2kg * hr2s;
    HR = SFC*LHV * 1/BTUlb2kJkg;
```

T–s diagram

```
    hMain = []; % placeholder in case no plot happens
if isa(TSplot, 'matlab.graphics.axis.Axes')
    ax = TSplot;  % Existing axes handle passed in
    hold(ax, 'on');
elseif isequal(TSplot, 1)
    figure;
    ax = gca;
    hold(ax, 'on');
else
    return;
end

% --- Define states
states = [1 2 25 3 4 48 5 6]';

% Reference for real entropy change (T-S diagram)
```

```matlab
sbref = 0;

% --- calculate specific entropies
sb1r = sbref;
sb2r   = sb1r + sb2 - sb1 - Rbar*log(p2/p1);
sb25r  = sb2r + sb25 - sb2 - Rbar*log(p25/p2);
sb25rs = sb2r + sb25s - sb2 - Rbar*log(p25/p2);
sb3rs  = sb25r + sb3s - sb25 - Rbar*log(p3/p25);
sb3r   = sb25r + sb3 - sb25 - Rbar*log(p3/p25);
sb4r   = sb3r + sb4 - sb3 - Rbar*log(p4/p3);
sb48r  = sb4r + sb48 - sb4 - Rbar*log(p48/p4);
sb48rs = sb4r + sb48s - sb4 - Rbar*log(p48/p4);
sb5r   = sb48r + sb5 - sb48 - Rbar*log(p5/p48);
sb5rs  = sb48r;
sb6r   = sb5r + sb6 - sb5 - Rbar*log(p6/p5);

% --- Collect specific entropies (kJ/kg·K)
allS = [ ...
    sb1r/M
    sb2r/M
    sb25r/M
    sb3r/M
    sb4r/M
    sb48r/M
    sb5r/M
    sb6r/M ];

% --- Assign a unique color from the current color order
colorOrder = ax.ColorOrder;
colorIndex = mod(ax.ColorOrderIndex-1, size(colorOrder,1)) + 1;
color = colorOrder(colorIndex, :);

% --- Plot 1-2-25-3
hMain = plot(ax, allS(1:4), [T1 T2 T25 T3], 'o-', ...
    'Color', color, 'LineWidth', 1, 'MarkerSize', 6, ...
    'DisplayName', sprintf('%.0f%% fuel rate', round(100*mdotf/1.8374)));



% --- Plot 4-48-5-6
plot(ax, allS(5:8), [T4 T48 T5 T6], 'o-', 'Color', color, ...
    'LineWidth', 1, 'MarkerSize', 6);

xlabel(ax, 'Specific Entropy s (kJ/kg·K)');
ylabel(ax, 'Temperature T (K)');
title(ax, ['Gas Turbine T-s Diagram at fuel mass flow: ', ...
           num2str(round(100*mdotf/1.8374)), '%']);
grid(ax, 'on');

% --- Label states
text(ax, allS, [T1 T2 T25 T3 T4 T48 T5 T6], string(states), ...
    'VerticalAlignment','bottom', 'HorizontalAlignment','right');

% --- Isentropic points
S_iso = [sb25rs/M, sb3rs/M, sb48rs/M, sb5rs/M];
T_iso = [T25s,     T3s,     T48s,    T5s];
labels_iso = {'25s','3s','48s','5s'};

scatter(ax, S_iso, T_iso, 60, [0.5 0.5 0.5], 'x', 'LineWidth', 1.5);
text(ax, S_iso, T_iso, labels_iso, ...
    'VerticalAlignment','top', 'HorizontalAlignment','left', ...
    'Color', [0.5 0.5 0.5],'FontSize',6);

% --- Connect turbine inlets to isentropic outlets
S_in = [sb2r/M, sb25r/M, sb4r/M, sb48r/M];
T_in = [T2,     T25,     T4,     T48];
```

```
for k = 1:length(S_iso)
    plot(ax, [S_iso(k) S_iso(k)], [T_in(k) T_iso(k)], ...
        '--', 'Color', [0.5 0.5 0.5], 'LineWidth', 0.8);
end

% --- Example isobars
iso_curve_25s_25 = ts_isobar(p25, T1, T25, p1, Rbar, T25s);
plot(ax, iso_curve_25s_25(:,1)/M, iso_curve_25s_25(:,2), ...
    '--', 'Color', [0.5 0.5 0.5], 'LineWidth', 1);

iso_curve_3s_3 = ts_isobar(p3, T1, T3, p1, Rbar, T3s);
plot(ax, iso_curve_3s_3(:,1)/M, iso_curve_3s_3(:,2), ...
    '--', 'Color', [0.5 0.5 0.5], 'LineWidth', 1);

iso_curve_3_4 = ts_isobar(p3, T1, T4, p1, Rbar, T3);
plot(ax, iso_curve_3_4(:,1)/M, iso_curve_3_4(:,2), ...
    '-', 'Color', color, 'LineWidth', 1);

iso_curve_48s_48 = ts_isobar(p48, T1, T48, p1, Rbar, T48s);
plot(ax, iso_curve_48s_48(:,1)/M, iso_curve_48s_48(:,2), ...
    '--', 'Color', [0.5 0.5 0.5], 'LineWidth', 1);

iso_curve_5s_5 = ts_isobar(p5, T1, T5, p1, Rbar, T5s);
plot(ax, iso_curve_5s_5(:,1)/M, iso_curve_5s_5(:,2), ...
    '--', 'Color', [0.5 0.5 0.5], 'LineWidth', 1);

iso_curve_1_6 = ts_isobar(p6, T1, T6, p1, Rbar, T1);
plot(ax, iso_curve_1_6(:,1)/M, iso_curve_1_6(:,2), ...
    '--', 'Color', color, 'LineWidth', 1);

hold(ax, 'off');
```

```
end
```