

**ALLEGATO 1:**  
**CODICE MATLAB PER IL CALCOLO DELLA**  
**PRESSIONE PER IL FLUSSO DI CO<sub>2</sub>**

```

1 %% CODICE PER IL CALCOLO DELLA PRESSIONE PER IL FLUSSO DI CO2
2
3 % Temperatura espressa in Kelvin
4 % Pressione espressa in Pascal
5 %
6 % stato liquido : modello fluido incomprimibile (se le condizioni supercritiche
7 % sono raggiunte attraversando la curva di saturazione, il fluido
8 % supercritico è trattato come incomprimibile)
9 % stato gas & supercritico: modello gas comprimibile con fattore di comprimibilità Z
10 % primo nodo testa pozzo, ultimo nodo sbocco in giacimento
11 % imponendo una pressione, temperatura e portata il codice calcola:
12 % evoluzione termodinamica nel pozzo, pressione temperatura e velocità,
13 % proprietà di interesse e altezza cambio fase
14 %
15 % caricamento proprietà in matrici nella forma:
16 % p[MPa] 0,3 . # 148 colonne . 15
17 % T[K]
18 % 280
19 % .
20 % # 41
21 % righe
22 % .
23 % 320
24 %
25 % Codice sviluppato da Francesco Pertuso
26 %
27
28 clear; clc; close all;
29 load('cp_CO2.mat')
30 load('density_CO2.mat')
31 load('th_conductivity_CO2.mat')
32 load('viscosity_CO2.mat')
33 load('viscosity_H2O.mat')
34 %% Input data
35 z = 2021; % [m] profondità pozzo
36 r_in = 0.0482; % [m] raggio interno
37 r_out = 0.0571; % [m] raggio esterno
38 r_ce = 0.24; % [m] raggio casing cemento
39 D = 2*r_in; % [m] diametro interno
40 A = pi*r_in^2; % [m^2] sezione interna
41 eps = 0.045e-3; % [m] rugosità
42 g = 9.81; % [m/s^2] costante di gravità
43 k_ss = 50; % [W/m/K] conducibilità acciaio
44 k_ce = 2.30; % [W/m/K] conducibilità cemento
45 R = 8.314462618; % [J/mol/K] costante universale dai gas
46 M = 0.0440095; % [kg/mol] massa molare CO2
47 p_crit = 73.8e5; % [Pa] pressione critica
48 T_crit = 30.97+273.15; % [K] temperatura critica
49 a = -0.015; % [K/m] gradiente geotermico
50 c = 288.45-0.615; % [K] temperatura sul fondale marino - 41m per scambio convettivo
51 Tgeo = @(z) c - a*z; % [K] z=0 profondità reservoir
52 % figure
53 % plot(Tgeo(z),z)
54 % title('geothermal gradient')
55 % grid on
56 % ylabel('well depth [m] -from surface to bottom-')
57 % xlabel('temperature [K]')
58 % set(gca, 'YDir','reverse')
59 %% Proprietà dell'acqua @ T = 16°C p = 1-2 bar (circa costanti)
60 T_H2O = 16 + 273.15; % [K] temperatura
61 rho_H2O = 998.95; % [kg/m3] densità
62 cp_H2O = 4187.4; % [J/kg/K] calore specifico
63 mu_H2O = 0.0011078; % [Pa*s] viscosità
64 k_th_H2O = 0.59077; % [W/m/K] conducibilità termica
65 u_H2O = 0.8; % [m/s] velocità
66 Re_H2O = rho_H2O*u_H2O*2*r_out/mu_H2O; % numero di Reynolds
67 Pr_H2O = cp_H2O*mu_H2O/k_th_H2O; % numero di Prandlt
68 %% Property dell'aria @ T = 15°C p = 1 bar
69 T_air = 15 + 273.15; % [K] temperatura
70 rho_air = 1.225; % [kg/m3] densità
71 cp_air = 1004.3; % [J/kg/K] calore specifico
72 mu_air = 1.781e-5; % [Pa*s] viscosità
73 k_th_air = 25.5e-3; % [W/m/K] conducibilità termica
74 u_air = 5; % [m/s] velocità
75 Re_air = rho_air*u_air*2*r_out/mu_air; % numero di Reynolds
76 Pr_air = cp_air*mu_air/k_th_air; % numero di Prandlt
77 %% Discretizzazione dominio computazionale
78 dz = 1;
79 zz = (0:dz:z)';
80 S = 2*pi*dz*r_in; % [m^2] superficie interna di scambio termico
81 % preallocazione vettori
82 p = zeros(size(zz));
83 T = zeros(size(zz));
84 v = zeros(size(zz));
85 rho = zeros(size(zz));
86 cp = zeros(size(zz));
87 mu = zeros(size(zz));

```

```

88 k_th = zeros(size(zz));
89 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%condizioni iniziali%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
90 m = -; % [kg/s] portata in massa
91 p_in = - + 1000; % [Pa] pressione iniziale + 0.001 MPa per interpolazione
92 T_in = - + 273.15; % [K] temperatura iniziale
93 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
94 %% Condizioni primo nodo
95 p(1) = p_in;
96 T(1) = T_in;
97 v(1) = m/C02_density(p(1),T(1),density_C02_m)/A;
98 i_change = 0; % ipotesi stato di partenza gassoso, viene corretto se ipotesi sbagliata
99 % controllo sullo stato termodinamico
100 if T(1) < T_crit && p(1) > C02_saturation_pressure(T(1))
101     incompressible = 1;
102     i_change = 1;
103     'stato di partenza liquido' %#ok<NOPTS>
104 else
105     incompressible = 0;
106     'stato di partenza gassoso' %#ok<NOPTS>
107 end
108 first = 1; % flag per prima iterazione di U
109 for ii=1:length(zz)-1
110     % calcolo proprietà per il nodo ii
111     rho(ii) = C02_density(p(ii),T(ii),density_C02_m);
112     cp(ii) = C02_cp(p(ii),T(ii),cp_C02_m);
113     mu(ii) = C02_viscosity(p(ii),T(ii),viscosity_C02_m);
114     k_th(ii) = C02_th_conductivity(p(ii),T(ii),th_conductivity_C02_m);
115     % modello fluido incompressibile Ma<0.3
116     if incompressible == 1
117         % numero di Reynolds
118         Re = m*D/mu(ii)/A;
119         % fattore di attrito
120         f = (-1.8*log((eps/2/r_in/3.78)^1.11 + 6.9/Re))^(0.5);
121         % equazione pressione -dalla conservazione del momento-
122         p(ii+1) = p(ii) + rho(ii)*g*dz - f*dz/D*m^2/rho(ii)/A^2;
123         if T(ii) <= T_crit && p(ii+1) <= C02_saturation_pressure(T(ii))
124             warning('cambiamento di fase da liquido a gas')
125             incompressible = 0;
126             i_change_2 = ii;
127         end
128     % modello gas reale
129     else
130         % fattore di comprimibilità
131         Z = p(ii)/rho(ii)/R*M/T(ii);
132         % numero di Reynolds
133         Re = m*D/mu(ii)/A;
134         % fattore di attrito
135         f = (-1.8*log((eps/D/3.78)^1.11 + 6.9/Re))^(0.5);
136         % equazione pressione -dalla conservazione del momento-
137         p(ii+1) = p(ii) + (dz*g*p(ii)*M/R/T(ii)/Z - dz*f*m^2*R*T(ii)/4/r_in*Z/p(ii)/M)...
138         /(1-m^2*R*T(ii)*Z/M/p(ii)^2);
139     end
140     if p(ii) < p_crit % usare correlazione Dittus Boelter correlation
141         % numero di Prandtl
142         Pr = cp(ii)*mu(ii)/k_th(ii);
143         % controllo validità correlazione Dittus-Boelter
144         if Pr < 0.6 || Pr > 160
145             error('numero di Prandtl fuori dal range di validità della correlazione di Dittus Boelter')
146         else
147             if Re < 1e4
148                 error('numero di Reynolds fuori dal range di validità della correlazione di Dittus Boelter')
149             end
150         end
151         % numero di Nusselt -dalla correlazione di Dittus Boelter-
152         Nu = 0.0243 * Re^(4/5) * Pr^(0.4);
153         % coefficiente di scambio termico convettivo interno
154         h_in = Nu*k_th(ii)/D;
155     else % usare correlazione C02 supercritica di Gupta
156         % calcolo il flusso termico con il coefficiente U dello step precedente
157         phi = (T(ii)-Tgeo(zz(ii)))*U;
158         % calcolo la temperatura a parete attraverso l'analogia elettrica
159         T_wall = T(ii) - phi/h_in;
160         % calcolo le proprietà con la temperatura a parete
161         k_th_wall = C02_th_conductivity(p(ii),T_wall,th_conductivity_C02_m);
162         mu_wall = C02_viscosity(p(ii),T_wall,viscosity_C02_m);
163         rho_wall = C02_density(p(ii),T_wall,density_C02_m);
164         cp_wall = C02_cp(p(ii),T_wall,cp_C02_m);
165         Pr_wall = cp_wall*mu_wall/k_th_wall;
166         Re_wall = m*D/mu_wall/A;
167         % numero di Nusselt -dalla correlazione di Gupta et al.-
168         Nu = 0.0038 * Re_wall^0.96 * Pr_wall^-0.14 * (rho_wall/rho(ii))^0.84 ...
169         * (k_th_wall/k_th(ii))^0.75 *(mu_wall/mu(ii))^-0.22;
170         % coefficiente di scambio termico convettivo interno
171         h_in = Nu*k_th(ii)/D;
172     end
173     % scambio termico convettivo esterno,
174     % nei primi 41 metri il pozzo non è a contatto con la crosta terrestre

```

```

175 if zz(ii) < 41
176     if zz(ii) <= 21 % fluido esterno: aria
177         if first == 1
178             % at first step wall and bulk temperature are the same
179             U = 0;
180         end
181         % temperatura a parete dalla conservazione radiale del flusso
182         T_w = T(ii) - U*S*(T(ii)-T_air)*log(r_out/r_in)/(2*pi*dz*k_ss);
183         mu_w = air_viscosity(T_w);
184         % numero di Nusselt -dalla correlazione di E.M.Sparrow et al.-
185         Nu1 = 0.025 + (0.4*Re_air^(1/2) + 0.06*Re_air^(2/3))*Pr_air^(0.37)*(mu_air/mu_w)^(1/4);
186         % coefficiente convezione esterna per l'aria
187         h_out = Nu1*k_th_air/D;
188         % coefficiente di scambio termico globale -convezione & conduzione & convezione-
189         U = 1/(1/h_in+r_in/k_ss*log(r_out/r_in)+r_in/r_out/h_out);
190         % equazione temperatura -dall'equazione del calore, pure advection problem-
191         T(ii+1) = T(ii) + S*U/m/cp(ii)*(T_air - T(ii));
192     else
193         % fluido estrno: acqua
194         % temperatura a parete dalla conservazione radiale del flusso
195         T_w = T(ii) - U*S*(T(ii)-T_H2O)*log(r_out/r_in)/(2*pi*dz*k_ss);
196         mu_w = H2O_viscosity(T_w,viscosity_H2O);
197         % numero di Nusselt -dalla correlazione di E.M.Sparrow et al.-
198         Nu1 = 0.025 + (0.4*Re_H2O^(1/2) + 0.06*Re_H2O^(2/3))*Pr_H2O^(0.37)*(mu_H2O/mu_w)^(1/4);
199         % coefficiente convezione esterna per l'acqua
200         h_out = Nu1*k_th_H2O/D;
201         % coefficiente di scambio termico globale -convezione & conduzione & convezione-
202         U = 1e-2*(1/h_in+r_in/k_ss*log(r_out/r_in)+r_in/r_out/h_out);
203         % equazione temperatura -dall'equazione del calore, pure advection problem-
204         T(ii+1) = T(ii) + S*U/m/cp(ii)*(T_H2O - T(ii));
205     end
206 else
207     % scambio termico conduttivo esterno,
208     % pozzo a contatto con la crosta terrestre
209     % coefficiente di scambio termico globale -convezione & conduzione-
210     U = 1/(1/h_in+r_in/k_ss*log(r_out/r_in)+r_out/k_ce*log(r_ce/r_out));
211     % equazione temperatura -dall'equazione del calore, constant surface temperature-
212     T(ii+1) = T(ii) + S*U/m/cp(ii)*(Tgeo(zz(ii))-T(ii));
213 end
214 % equazione della velocità -dalla continuità-
215 v(ii+1) = m/rho(ii)/A;
216 % controllo massima pressione del reservoir
217 if p(ii+1) > 145.1e5
218     strcat(['massima pressione del reservoir raggiunta nel pozzo a \' ...
219           '\',num2str(2000 - zz(ii+1)),' metri dallo sbocco'])
220     ii = ii+1; %ok<FXSET>
221     break
222 end
223 % controllo cambiamento di fase gas->liquido
224 if i_change == 0
225     if T(ii+1) < T_crit && p(ii+1) >= C02_saturation_pressure(T(ii+1))
226         incompressible = 1;
227         i_change = ii;
228         warning('cambiamento di fase da gas a liquido')
229     else
230         % controllo cambiamento di fase sup_gas->sup_liquido
231         if p(ii+1) >= C02_saturation_pressure(T(ii+1))
232             i_change = ii;
233             warning('cambiamento di fase da gas supercritico a liquido supercritico')
234         end
235     end
236 end
237 end
238
239 %% Plot section
240 % set defaults for figures
241 set(0,'defaultlinelinerwidth',2)
242 set(0,'defaultaxesfontsize',20)
243 set(0,'defaulttextfontsize',20)
244 set(0,'defaultlegendautoupdate','off')
245 close all
246 strcat('pressione finale:',num2str(p(ii)/1e5),' bar')
247 %% Evoluzione termodinamica nel pozzo
248 temp = 217:0.1:T_crit; % [K]
249 temp_1 = T_crit:0.1:340;
250 p_sat = zeros(size(temp));
251 p_pscr = zeros(size(temp_1));
252 for i_sat = 1:length(temp)
253     p_sat(i_sat) = C02_saturation_pressure(temp(i_sat));
254 end
255 for i_sat = 1:length(temp_1)
256     p_pscr(i_sat) = C02_saturation_pressure(temp_1(i_sat));
257 end
258 figure
259 tiledlayout('flow')
260 if i_change == 0 % senza cambiamento di fase
261     nexttile

```

```

262 hold on
263 plot(T(1:ii)-273.15,p(1:ii)/1e5)
264 plot(temp-273.15,p_sat/1e5,'k-',temp_1-273.15,p_pscr/1e5,'k-.')
265 plot([T_crit-273.15,400-273.15],[p_crit,p_crit]./1e5,'k--')
266 plot([T_crit-273.15,T_crit-273.15],[p_crit/1e5,145],'k--')
267 title('EVOLUZIONE TERMODINAMICA')
268 xlabel('Temperatura [°C]')
269 ylabel('Pressione [bar]')
270 legend('evoluzione termodinamica','curva saturazione','curva pseudocritica' ...
271         , 'condizioni supercritiche','Location','best')
272 grid on
273 xlim([273-273.15,400-273.15])
274 ylim([20,145])
275 %% Evoluzione della pressione nel pozzo
276 nexttile
277 plot(p(1:ii)./1e5,zz(1:ii))
278 title('EVOLUZIONE PRESSIONE')
279 xlabel('Pressione [bar]')
280 ylabel('Profondità pozzo [m]')
281 box on
282 grid minor
283 set(gca, 'YDir','reverse')
284 %% Evoluzione della temperatura nel pozzo
285 nexttile
286 plot(T(1:ii)-273.15,zz(1:ii),Tgeo(zz(41/dz:ii))-273.15,zz(41/dz:ii))
287 box on
288 title('EVOLUZIONE TEMPERATURA')
289 xlabel('Temperatura [°C]')
290 ylabel('Profondità pozzo [m]')
291 legend('Temperatura fluido','Temperatura crosta terrestre','Location','best')
292 grid minor
293 set(gca, 'YDir','reverse')
294 %% Evoluzione velocità nel pozzo
295 nexttile
296 plot(v(1:ii),zz(1:ii))
297 title('EVOLUZIONE VELOCITÀ')
298 xlabel('Velocità [m/s]')
299 ylabel('Profondità pozzo [m]')
300 grid minor
301 box on
302 set(gca, 'YDir','reverse')
303 %% Evoluzione proprietà nel pozzo
304 figure
305 tiledlayout('flow')
306 % densità
307 nexttile
308 plot(rho(1:ii),zz(1:ii))
309 title('EVOLUZIONE DENSITÀ')
310 xlabel('ρ [kg/m^3]')
311 ylabel('Profondità pozzo [m]')
312 grid minor
313 set(gca, 'YDir','reverse')
314 % viscosità
315 nexttile
316 plot(mu(1:ii),zz(1:ii))
317 title('EVOLUZIONE VISCOSITÀ')
318 xlabel('μ [Pa*s]')
319 ylabel('Profondità pozzo [m]')
320 grid minor
321 box on
322 set(gca, 'YDir','reverse')
323 % calore specifico
324 nexttile
325 plot(cp(1:ii),zz(1:ii))
326 title('EVOLUZIONE CALORE SPECIFICO')
327 xlabel('c_p [J/Kg/K]')
328 ylabel('Profondità pozzo [m]')
329 grid minor
330 box on
331 set(gca, 'YDir','reverse')
332 % conducibilità termica
333 nexttile
334 plot(k_th(1:ii),zz(1:ii))
335 title('EVOLUZIONE CONDUCIBILITÀ TERMICA')
336 xlabel('k [W/m/K]')
337 ylabel('Profondità pozzo [m]')
338 grid minor
339 box on
340 set(gca, 'YDir','reverse')
341 else % con cambiamento di fase
342 strcat('altezza cambio fase:', num2str(zz(i_change)),' m')
343 nexttile
344 hold on
345 %% Evoluzione termodinamica nel pozzo
346 plot(T(1:ii)-273.15,p(1:ii)/1e5,T(i_change)-273.15,p(i_change)/1e5,'x')
347 plot(temp-273.15,p_sat/1e5,'k-',temp_1-273.15,p_pscr/1e5,'k-.')
348 plot([T_crit-273.15,400-273.15],[p_crit,p_crit]./1e5,'k--')

```

```

349 plot([T_crit-273.15,T_crit-273.15],[p_crit/1e5,145],'k--')
350 title('EVOLUZIONE TERMODINAMICA')
351 xlabel('Temperatura [°C]')
352 ylabel('Pressione [bar]')
353 legend('evoluzione termodinamica','passaggio da gas-like a liquid-like',['curva ' ...
354 'saturazione'],'curva pseudocritica','condizioni supercritiche','Location','best')
355 grid on
356 box on
357 xlim([273-273.15,400-273.15])
358 ylim([20,145])
359 %% Evoluzione della pressione nel pozzo
360 nexttile
361 plot(p(1:ii)./1e5,zz(1:ii),p(i_change)/1e5,zz(i_change),'x')
362 title('EVOLUZIONE PRESSIONE')
363 xlabel('Pressione [bar]')
364 ylabel('Profondità pozzo [m]')
365 legend('','passaggio da gas-like a liquid-like','Location','best')
366 grid minor
367 box on
368 set(gca, 'YDir','reverse')
369 %% Evoluzione della temperatura nel pozzo
370 nexttile
371 plot(T(1:ii)-273.15,zz(1:ii),Tgeo(zz(41/dz:ii))-273.15,zz(41/dz:ii),T(i_change)-273.15,zz(i_change),'x')
372 title('EVOLUZIONE TEMPERATURA')
373 xlabel('Temperatura [°C]')
374 ylabel('Profondità pozzo [m]')
375 legend('Temperatura fluido','Temperatura crosta terrestre',['passaggio da gas-like a ' ...
376 'liquid-like'],'Location','best')
377 grid minor
378 box on
379 set(gca, 'YDir','reverse')
380 %% Evoluzione della velocità nel pozzo
381 nexttile
382 plot(v(1:ii),zz(1:ii),v(i_change),zz(i_change),'x')
383 title('EVOLUZIONE VELOCITÀ')
384 xlabel('Velocità [m/s]')
385 ylabel('Profondità pozzo [m]')
386 legend('','passaggio da gas-like a liquid-like','Location','best')
387 grid minor
388 box on
389 set(gca, 'YDir','reverse')
390 %% Evoluzione proprietà nel pozzo
391 figure
392 tiledlayout('flow')
393 % densità
394 nexttile
395 plot(rho(1:ii),zz(1:ii),rho(i_change),zz(i_change),'x')
396 title('EVOLUZIONE DENSITÀ')
397 xlabel('p [kg/m^3]')
398 ylabel('Profondità pozzo [m]')
399 legend('','passaggio da gas-like a liquid-like','Location','best')
400 grid minor
401 box on
402 set(gca, 'YDir','reverse')
403 % viscosità
404 nexttile
405 plot(mu(1:ii),zz(1:ii),mu(i_change),zz(i_change),'x')
406 title('EVOLUZIONE VISCOSITÀ')
407 xlabel('μ [Pa*s]')
408 ylabel('Profondità pozzo [m]')
409 legend('','passaggio da gas-like a liquid-like','Location','best')
410 grid minor
411 box on
412 set(gca, 'YDir','reverse')
413 % calore specifico
414 nexttile
415 plot(cp(1:ii),zz(1:ii),cp(i_change),zz(i_change),'x')
416 title('EVOLUZIONE CALORE SPECIFICO')
417 xlabel('c_p [J/Kg/K]')
418 ylabel('Profondità pozzo [m]')
419 legend('','passaggio da gas-like a liquid-like','Location','best')
420 grid minor
421 box on
422 set(gca, 'YDir','reverse')
423 % conducibilità termica
424 nexttile
425 plot(k_th(1:ii),zz(1:ii),k_th(i_change),zz(i_change),'x')
426 title('EVOLUZIONE CONDUCIBILITÀ TERMICA')
427 xlabel('k [W/m/K]')
428 ylabel('Profondità pozzo [m]')
429 legend('','passaggio da gas-like a liquid-like','Location','best')
430 grid minor
431 box on
432 set(gca, 'YDir','reverse')
433 end
434 figure(1)
435

```

```

436 %% FUNZIONI PROPRIETA' TERMODINAMICHE CO2
437
438 function density = CO2_density(p,T,density_CO2_m)
439 % questa funzione interpola linearmente il valore di densità
440 % sul piano pressione-temperatura
441 % range pressione: 3 < p [MPa] < 15
442 %                 30 < p [bar] < 150
443 % discretizzazione 0.02 MPa
444 %
445 % range temperatura: 7 < T [°C] < 127
446 %                 280 < T [K] < 400
447 % discretizzazione 0.5 K
448 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
449 % Questo script lavora con pressione in MPa e temperatura in K %
450 % è necessario passare le variabili alla funzione in Pa e K %
451 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
452 % basato sul database NIST
453
454 % controllo parametri di input
455 if T < 280 || T > 400
456     error('Temperature out of boundaries')
457 end
458 if p < 3e6 || p > 15e6
459     error('pressure out of boundaries')
460 end
461 % conversione Pa in MPa
462 p = 1e-6 * p;
463 % trovare i valori di pressione di riferimento nella griglia discretizzata con 0.02 MPa
464 % example p = 3.023 MPa
465 p = p*100; % 302.3
466 p_floor = floor(p)/100; % 3.020
467 if mod(p_floor,0.02) ~= 0 % mod = 0
468     p_floor = p_floor - mod(p_floor,0.02);
469 end
470 p_ceil = ceil(p)/100;
471 if mod(p_ceil,0.02) ~= 0 % mod = 0.01
472     p_ceil = p_ceil + 0.02-mod(p_ceil,0.02); % 3.04end
473 end
474 % riportare la pressione in MPa
475 p = p/100;
476 % trovare i valori di temperatura di riferimento nella griglia discretizzata con 0.5 K
477 % example T = 290.5832 K
478 T = 10*T; % 2905.832
479 T_floor = floor(T)/10; % 290.5000
480 if mod(T_floor,0.5) ~= 0 % mod = 0
481     T_floor = T_floor - mod(T_floor,0.5);
482 end
483 T_ceil = ceil(T)/10; % 290.6000
484 if mod(T_ceil,0.5) ~= 0 % mod = 0.1
485     T_ceil = T_ceil + 0.5-mod(T_ceil,0.5); % 291.0
486 end
487 % riportare temperatura in K
488 T = T/10;
489 % X->p [Mpa] Y->T [K] riferimento
490 X = [p_floor; p_ceil];
491 Y = [T_floor; T_ceil];
492 % riporto i valori numerici di pressione e temperatura in indici nelle
493 % parentesi ed estraggo dalla matrice di riferimento i valori di riferimento
494 p_T_floor = density_CO2_m(int16(1+(T_floor-280)*2),int16(1+(p_floor-3)*50));
495 p_ceil_T_floor = density_CO2_m(int16(1+(T_floor-280)*2),int16(1+(p_ceil-3)*50));
496 p_floor_T_ceil = density_CO2_m(int16(1+(T_ceil-280)*2),int16(1+(p_floor-3)*50));
497 p_T_ceil = density_CO2_m(int16(1+(T_ceil-280)*2),int16(1+(p_ceil-3)*50));
498 % V è la matrice con i valori di output in X,Y
499 V = [p_T_floor, p_ceil_T_floor; p_floor_T_ceil, p_T_ceil];
500
501 density = interp2(X,Y,V,p,T,'linear');
502 end
503
504 function cp = CO2_cp(p,T,cp_CO2_m)
505 % questa funzione interpola linearmente il valore del calore
506 % specifico a pressione costante sul piano pressione-temperatura
507 % range pressione: 3 < p [MPa] < 15
508 %                 30 < p [bar] < 150
509 % discretizzazione 0.02 MPa
510 %
511 % range temperatura: 7 < T [°C] < 127
512 %                 280 < T [K] < 400
513 % discretizzazione 0.5 K
514 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
515 % Questo script lavora con pressione in MPa e temperatura in K %
516 % è necessario passare le variabili alla funzione in Pa e K %
517 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
518 % basato sul database NIST
519
520 % controllo parametri di input
521 if T < 280 || T > 400
522     error('Temperature out of boundaries')

```

```

523 end
524 if p < 3e6 || p > 15e6
525     error('pressure out of boundaries')
526 end
527 % conversione Pa in MPa
528 p = 1e-6 * p;
529 % trovare i valori di pressione di riferimento nella griglia discretizzata con 0.02 MPa
530                                     % example p = 3.023 MPa
531 p = p*100;                           % 302.3
532 p_floor = floor(p)/100;               % 3.020
533 if mod(p_floor,0.02) ~= 0             % mod = 0
534     p_floor = p_floor - mod(p_floor,0.02);
535 end
536 p_ceil = ceil(p)/100;
537 if mod(p_ceil,0.02) ~= 0               % mod = 0.01
538     p_ceil = p_ceil + 0.02-mod(p_ceil,0.02); % 3.04
539 end
540 % riportare la pressione in MPa
541 p = p/100;
542 % trovare i valori di temperatura di riferimento nella griglia discretizzata con 0.5 K
543                                     % example T = 290.5832 K
544 T = 10*T;                             % 2905.832
545 T_floor = floor(T)/10;                 % 290.5000
546 if mod(T_floor,0.5) ~= 0              % mod = 0
547     T_floor = T_floor - mod(T_floor,0.5);
548 end
549 T_ceil = ceil(T)/10;                   % 290.6000
550 if mod(T_ceil,0.5) ~= 0                % mod = 0.1
551     T_ceil = T_ceil + 0.5-mod(T_ceil,0.5); % 291.0
552 end
553 % riportare temperatura in K
554 T = T/10;
555 % X->p [Mpa] Y->T [K] riferimento
556 X = [p_floor; p_ceil];
557 Y = [T_floor; T_ceil];
558 % riporto i valori numerici di pressione e temperatura in indici nelle
559 % parentesi ed estraggo dalla matrice di riferimento i valori di riferimento
560 p_T_floor = cp_CO2_m(int16(1+(T_floor-280)*2),int16(1+(p_floor-3)*50));
561 p_ceil_T_floor = cp_CO2_m(int16(1+(T_floor-280)*2),int16(1+(p_ceil-3)*50));
562 p_floor_T_ceil = cp_CO2_m(int16(1+(T_ceil-280)*2),int16(1+(p_floor-3)*50));
563 p_T_ceil = cp_CO2_m(int16(1+(T_ceil-280)*2),int16(1+(p_ceil-3)*50));
564 % V è la matrice con i valori di output in X,Y
565 V = [p_T_floor, p_ceil_T_floor; p_floor_T_ceil, p_T_ceil];
566
567 cp = interp2(X,Y,V,p,T,'linear');
568 end
569
570 function th_conductivity = CO2_th_conductivity(p,T,th_conductivity_CO2_m)
571 % questa funzione interpola linearmente il valore della conducibilità
572 % termica sul piano pressione-temperatura
573 % range pressione: 3 < p [MPa] < 15
574 %                 30 < p [bar] < 150
575 % discretizzazione 0.02 MPa
576 %
577 % range temperatura: 7 < T [°C] < 127
578 %                  280 < T [K] < 400
579 % discretizzazione 0.5 K
580 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
581 % Questo script lavora con pressione in MPa e temperatura in K %
582 % è necessario passare le variabili alla funzione in Pa e K %
583 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
584 % basato sul database NIST
585
586 % controllo parametri di input
587 if T < 280 || T > 400
588     error('Temperature out of boundaries')
589 end
590
591 if p < 3e6 || p > 15e6
592     error('pressure out of boundaries')
593 end
594 % conversione Pa in MPa
595 p = 1e-6 * p;
596 % trovare i valori di pressione di riferimento nella griglia discretizzata con 0.02 MPa
597                                     % example p = 3.023 MPa
598 p = p*100;                           % 302.3
599 p_floor = floor(p)/100;               % 3.020
600 if mod(p_floor,0.02) ~= 0             % mod = 0
601     p_floor = p_floor - mod(p_floor,0.02);
602 end
603 p_ceil = ceil(p)/100;
604 if mod(p_ceil,0.02) ~= 0               % mod = 0.01
605     p_ceil = p_ceil + 0.02-mod(p_ceil,0.02); % 3.04
606 end
607 % riportare la pressione in MPa
608 p = p/100;
609 % trovare i valori di temperatura di riferimento nella griglia discretizzata con 0.5 K

```

```

610 % example T = 290.5832 K
611 T = 10*T; % 2905.5832
612 T_floor = floor(T)/10; % 290.5000
613 if mod(T_floor,0.5) ~= 0 % mod = 0
614     T_floor = T_floor - mod(T_floor,0.5);
615 end
616 T_ceil = ceil(T)/10; % 290.6000
617 if mod(T_ceil,0.5) ~= 0 % mod = 0.1
618     T_ceil = T_ceil + 0.5-mod(T_ceil,0.5); % 291.0
619 end
620 % riportare temperatura in K
621 T = T/10;
622 % X->p [Mpa] Y->T [K] riferimento
623 X = [p_floor; p_ceil];
624 Y = [T_floor; T_ceil];
625 % riporto i valori numerici di pressione e temperatura in indici nelle
626 % parentesi ed estraggo dalla matrice di riferimento i valori di riferimento
627 p_T_floor = th_conductivity_CO2_m(int16(1+(T_floor-280)*2),int16(1+(p_floor-3)*50));
628 p_ceil_T_floor = th_conductivity_CO2_m(int16(1+(T_floor-280)*2),int16(1+(p_ceil-3)*50));
629 p_floor_T_ceil = th_conductivity_CO2_m(int16(1+(T_ceil-280)*2),int16(1+(p_floor-3)*50));
630 p_T_ceil = th_conductivity_CO2_m(int16(1+(T_ceil-280)*2),int16(1+(p_ceil-3)*50));
631 % V è la matrice con i valori di output in X,Y
632 V = [p_T_floor, p_ceil_T_floor; p_floor_T_ceil, p_T_ceil];
633
634 th_conductivity = interp2(X,Y,V,p,T,'linear');
635 end
636
637 function viscosity = CO2_viscosity(p,T,viscosity_CO2_m)
638 % questa funzione interpola linearmente il valore della viscosità
639 % sul piano pressione-temperatura
640 % range pressione: 3 < p [MPa] < 15
641 % 30 < p [bar] < 150
642 % discretizzazione 0.02 MPa
643 %
644 % range temperatura: 7 < T [°C] < 127
645 % 280 < T [K] < 400
646 % discretizzazione 0.5 K
647 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
648 % Questo script lavora con pressione in MPa e temperatura in K %
649 % è necessario passare le variabili alla funzione in Pa e K %
650 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
651 % basato sul database NIST
652
653 % controllo parametri di input
654 if T < 280 || T > 400
655     error('Temperature out of boundaries')
656 end
657 if p < 3e6 || p > 15e6
658     error('pressure out of boundaries')
659 end
660 % conversione Pa in MPa
661 p = 1e-6 * p;
662 % trovare i valori di pressione di riferimento nella griglia discretizzata con 0.02 MPa
663 % example p = 3.023 MPa
664 p = p*100; % 302.3
665 p_floor = floor(p)/100; % 3.020
666 if mod(p_floor,0.02) ~= 0 % mod = 0
667     p_floor = p_floor - mod(p_floor,0.02);
668 end
669 p_ceil = ceil(p)/100;
670 if mod(p_ceil,0.02) ~= 0 % mod = 0.01
671     p_ceil = p_ceil + 0.02-mod(p_ceil,0.02); % 3.04
672 end
673 % riportare la pressione in MPa
674 p = p/100;
675 % trovare i valori di temperatura di riferimento nella griglia discretizzata con 0.5 K
676 % example T = 290.5832 K
677 T = 10*T; % 2905.5832
678 T_floor = floor(T)/10; % 290.5000
679 if mod(T_floor,0.5) ~= 0 % mod = 0
680     T_floor = T_floor - mod(T_floor,0.5);
681 end
682 T_ceil = ceil(T)/10; % 290.6000
683 if mod(T_ceil,0.5) ~= 0 % mod = 0.1
684     T_ceil = T_ceil + 0.5-mod(T_ceil,0.5); % 291.0
685 end
686 % riportare temperatura in K
687 T = T/10;
688 % X->p [Mpa] Y->T [K] reference
689 X = [p_floor; p_ceil];
690 Y = [T_floor; T_ceil];
691 % riporto i valori numerici di pressione e temperatura in indici nelle
692 % parentesi ed estraggo dalla matrice di riferimento i valori di riferimento
693 p_T_floor = viscosity_CO2_m(int16(1+(T_floor-280)*2),int16(1+(p_floor-3)*50));
694 p_ceil_T_floor = viscosity_CO2_m(int16(1+(T_floor-280)*2),int16(1+(p_ceil-3)*50));
695 p_floor_T_ceil = viscosity_CO2_m(int16(1+(T_ceil-280)*2),int16(1+(p_floor-3)*50));
696 p_T_ceil = viscosity_CO2_m(int16(1+(T_ceil-280)*2),int16(1+(p_ceil-3)*50));

```

```

697 % V è la matrice con i valori di output in X,Y
698 V = [p_T_floor, p_ceil_T_floor; p_floor_T_ceil, p_T_ceil];
699
700 viscosity = interp2(X,Y,V,p,T,'linear');
701 end
702
703 function viscosity = air_viscosity(T)
704 % questa funzione calcola la viscosità dell'aria per una pressione
705 % p = 1 bar ed un range di temperatura: 100 < T [K] < 3000
706 %
707 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
708 % lo script lavora in Kelvin, passare la temperatura in Kelvin %
709 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
710 % basato sulla correlazione di A.I. Zografos et al.
711
712 if T < 100 || T > 3000
713     error('air temperature out of limits')
714 end
715 viscosity = 2.5914e-15 * T^3 - 1.4346e-11 * T^2 + 5.0523e-8 * T + 4.1130e-6;
716 end
717
718 function viscosity = H2O_viscosity(T,viscosity_H20)
719 % questa funzione calcola la viscosità dell'acqua per una pressione
720 % p = 1 bar ed un range di temperatura: 5 < T [°C] < 150
721 %                                     278 < T [K] < 423
722 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
723 % lo script lavora in Kelvin, passare la temperatura in Kelvin %
724 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
725 % basato sul database NIST
726
727 if T < 5+273 || T > 150+273
728     error('water temperature out of limits')
729 end
730 T = T - 273.15;
731 TT = 5:0.5:150;
732 viscosity = interp1(TT,viscosity_H20,T);
733 end
734
735 function p_sat = CO2_saturation_pressure(T)
736 % questa funzione calcola la pressione di saturazione per la CO2 per
737 % temperature maggiori a quella del punto triplo T = 216.59 K ed inferiori a quella
738 % critica T = 304.12 K
739 % temperatura output in Kelvin
740 %
741 % basato sulla correlazione del paper: "A New Equation of state for Carbon Dioxide"
742
743 if T < 216.59
744     error('temperature below triple point boundary')
745 % else
746 %     if T > 304.12
747 %         warning('temperature above critical point, pseudocritical line')
748 %     end
749 end
750 p_c = 7.3773e6; % [Pa] pressione critica
751 T_c = 304.1282; % [K] temperatura critica
752 a_1 = -7.0602087; a_2 = 1.9391218; a_3 = -1.6463597; a_4 = -3.2995634;
753 t_1 = 1.0; t_2 = 1.5; t_3 = 2.0; t_4 = 4.0;
754 p_sat = p_c * exp(T_c/T * ( a_1*(1-T/T_c)^t_1 + a_2*(1-T/T_c)^t_2 + a_3*(1-T/T_c)^t_3 + a_4*(1-T/T_c)^t_4));
755 p_sat = real(p_sat);
756 end

```