付録 A

Scilab ソースコード

A.1 斉次線型常微分方程式を解く Scilab スクリプト: exp_hom.sce

```
 1: clear();
 2:
 3: // relative error
 4: function rerr = relerr(approx_vec, true_vec)
 5:   tmp_vec = abs(approx_vec - true_vec);
 6:   for i = 1:1:lode_dim
 7:     tmp = abs(true_vec(i));
 8:     if tmp > 1.0e-13 then
 9:       tmp_vec(i) = tmp_vec(i) / tmp;
10:     end
11:   end
12:   rerr = max(tmp_vec);
13: endfunction
14:
15: function rerr = relerr_mat_diag(approx_mat, true_mat)
16:   tmp_mat = abs(approx_mat - true_mat);
17:   for i = 1:1:lode_dim
18:     tmp = abs(true_mat(i, i));
19:     if tmp > 1.0e-13 then
20:       tmp_mat(i, i) = tmp_mat(i, i) / tmp;
21:     end
22:   end
23:   rerr = max(tmp_mat);
24: endfunction
25:
```

```
26: // Problem
27: // y' = Ay + g(t)
28: // y(0) = y0
29: lode_dim = 20;
30:
31: // D = diag[-i]
32: org_D = eye(lode_dim, lode_dim);
33: for i = 1:1:lode_dim
34:   org_D(i, i) = -i;
35: end
36: rand('seed', 10);
37: V = rand(lode_dim, lode_dim);
38: inv_V = V^(-1);
39:
40: A = inv_V * org_D * V;
41:
42: //t = 5;
43:
44: for i = 1:1:lode_dim
45:   y0(i) = 1;
46: end
47:
48: function [vec] = g(t)
49:   for i = 1:1:lode_dim
50:     vec(i) = i * t;
51: //    vec(i) = 0;
52:   end
53: endfunction
54:
55: // value of t
56: //t_end = 1;
57: t_end = 10;
58:
59: // Exact Solution
60: function [vec] = sol(t)
61: //  vec = zeros(lode_dim, 1);
62:   exp_mat = eye(lode_dim, lode_dim);
63:   for i = 1:1:lode_dim
```

```
 64:     exp_mat(i, i) = exp(org_D(i, i) * t);
 65:   end
 66: //  vec = inv_V * (exp_mat * (V * y0));
 67:   vec = V * (exp_mat * (inv_V * y0));
 68: endfunction
 69:
 70: // for ODE solver
 71: function [ydot] = ode_func(t, y)
 72:   ydot = A * y;
 73: endfunction
 74: // End of Problem
 75:
 76: org_A = A;
 77: [V, D] = spec(A);
 78: abs_D = abs(diag(D));
 79: //inv_V = V^(-1);
 80: inv_V = inv(V);
 81: [l_V, u_V] = lu(V);
 82:
 83: printf("DIM = %d\n", lode_dim);
 84: printf("cond(V) = %f\n", cond(V));
 85: printf("max|lambda_i|, min|lambda_i| = %f, %f\n", max(abs_D), min(a
bs_D));
 86: printf("Cond(A) = %f, ||A||_2 = %f\n", norm(org_A, 2)*norm(inv(org_
A), 2), norm(A,2));
 87: //cond(A), norm(A, 2));
 88: printf("lambda_1, lambda_n = %f, %f\n", D(1, 1), D(lode_dim, lode_d
im));
 89: printf("relerr(lambda): %e\n", relerr(gsort(diag(D), 'g', 'i'), gso
rt(diag(org_D), 'g')));
 90:
 91: //return;
 92:
 93: // set torelances
 94: ode_rerr = 1.0e-10;
 95: ode_aerr = 0.0;
 96:
 97: // trial times
```

```
 98: max_trial_times = 5;
 99:
100: // Solver
101: function [vec] = hat_g(t)
102:   vec = inv_V * g(t);
103: endfunction
104:
105: hat_y0 = inv_V * y0;
106:
107: // Linear ODE solver (best case)
108: function [y] = best_lode_solver(mat, t_end, gfunc, init_y)
109:   lambda = spec(mat);
110:
111: //  printf("||A - V * lambda * inv_V|| / ||A||: %g\n", norm(mat - V
 * diag(lambda) * inv_V, 2) / norm(mat, 2));
112:
113:   init_y = inv_V * init_y;
114:
115:   for i = 1:1:lode_dim
116:     y(i) = exp(lambda(i) * t_end) * init_y(i);
117:   end
118:   y = y;
119:   y = V * y;
120: endfunction
121:
122: printf("Best_y\n");
123: total_best_y_time = 0;
124: for trial_times = 1:1:max_trial_times
125:   tic();
126:   best_y = best_lode_solver(A, t_end, hat_g, y0);
127:   best_y_time = toc();
128:   printf("best_y(%d/%d): %f (sec)\n", trial_times, max_trial_times,
 best_y_time);
129:   total_best_y_time = total_best_y_time + best_y_time;
130: end
131:
132: printf("ODE solver(default)\n");
133: total_ode_default_y_time = 0;
```

```
134: for trial_times = 1:1:max_trial_times;
135:   tic();
136:   ode_default_y = ode(y0, 0, t_end, ode_rerr, ode_aerr, ode_func);
137:   ode_default_y_time = toc();
138:   printf("ode_y_default_time(%d/%d): %f\n", trial_times, max_trial_
times, ode_default_y_time);
139:   total_ode_default_y_time = total_ode_default_y_time + ode_default
_y_time;
140: end
141:
142: printf("ODE solver(stiff)\n");
143: total_ode_stiff_y_time = 0;
144: for trial_times = 1:1:max_trial_times;
145:   tic();
146:   ode_stiff_y = ode("stiff", y0, 0, t_end, ode_rerr, ode_aerr, ode_
func);
147:   ode_stiff_y_time = toc();
148:   printf("ode_y_stiff_time(%d/%d): %f\n", trial_times, max_trial_ti
mes, ode_stiff_y_time);
149:   total_ode_stiff_y_time = total_ode_stiff_y_time + ode_stiff_y_tim
e;
150: end
151:
152: printf("Exact Solution:");
153: true_y = sol(t_end);
154:
155:
156: printf("time(sec): %f, %f, %f\n", total_best_y_time/max_trial_times
, total_ode_default_y_time/max_trial_times, total_ode_stiff_y_time/max_t
rial_times);
157: printf("Relerr(best_y): %e, %e\n", relerr(ode_default_y, best_y), r
elerr(ode_stiff_y, best_y));
158: printf("Relerr        : %e\n", relerr(best_y, true_y));
159: printf("y(1), y(n): %e, %e\n", best_y(1), best_y(lode_dim))
```

A.2 非斉次線型常微分方程式を解く Scilab スクリプト: exp_inhom.sce

```
 1: clear();
 2:
 3: // relative error
 4: function rerr = relerr(approx_vec, true_vec)
 5:   tmp_vec = abs(approx_vec - true_vec);
 6:   for i = 1:1:lode_dim
 7:     tmp = abs(true_vec(i));
 8:     if tmp > 1.0e-13 then
 9:       tmp_vec(i) = tmp_vec(i) / tmp;
10:     end
11:   end
12:   rerr = max(tmp_vec);
13: endfunction
14:
15: function rerr = relerr_mat_diag(approx_mat, true_mat)
16:   tmp_mat = abs(approx_mat - true_mat);
17:   for i = 1:1:lode_dim
18:     tmp = abs(true_mat(i, i));
19:     if tmp > 1.0e-13 then
20:       tmp_mat(i, i) = tmp_mat(i, i) / tmp;
21:     end
22:   end
23:   rerr = max(tmp_mat);
24: endfunction
25:
26: // Problem
27: // y' = Ay + g(t)
28: // y(0) = y0
29: //lode_dim = 5;
30: //lode_dim = 10;
31: lode_dim = 20;
32: //lode_dim = 30;
33: //lode_dim = 50;
```

```
34:
35: // D = diag[-i]
36: org_D = eye(lode_dim, lode_dim);
37: for i = 1:1:lode_dim
38:   org_D(i, i) = -i;
39: end
40: rand('seed', 10);
41: V = rand(lode_dim, lode_dim);
42: inv_V = V^(-1);
43:
44: A = inv_V * org_D * V;
45:
46: //t = 5;
47:
48: for i = 1:1:lode_dim
49:   y0(i) = 1;
50: end
51:
52: function [vec] = g(t)
53:   for i = 1:1:lode_dim
54:     vec(i) = i * t;
55: //    vec(i) = 0;
56:   end
57: endfunction
58:
59: // value of t
60: //t_end = 1;
61: t_end = 10;
62:
63: // for ODE solver
64: function [ydot] = ode_func(t, y)
65:   ydot = A * y + g(t);
66: endfunction
67: // End of Problem
68:
69: org_A = A;
70: [V, D] = spec(A);
71: abs_D = abs(diag(D));
```

```
 72: //inv_V = V^(-1);
 73: inv_V = inv(V);
 74: [l_V, u_V] = lu(V);
 75:
 76: printf("DIM = %d\n", lode_dim);
 77: printf("cond(V) = %f\n", cond(V));
 78: printf("max|lambda_i|, min|lambda_i| = %f, %f\n", max(abs_D), min(a
bs_D));
 79: printf("Cond(A) = %f, ||A||_2 = %f\n", norm(org_A, 2)*norm(inv(org_
A), 2), norm(A,2));
 80: //cond(A), norm(A, 2));
 81: printf("lambda_1, lambda_n = %f, %f\n", D(1, 1), D(lode_dim, lode_d
im));
 82: //gsort(diag(D), 'g', 'i'), gsort(diag(org_D), 'g')
 83: printf("relerr(lambda): %e\n", relerr(gsort(diag(D), 'g', 'i'), gso
rt(diag(org_D), 'g')));
 84:
 85: //return;
 86:
 87: // set torelances
 88: ode_rerr = 1.0e-10;
 89: //ode_rerr = 1.0e-5;
 90: ode_aerr = 0.0;
 91: intg_rerr = 1.0e-10;
 92: intg_aerr = 0.0;
 93:
 94: // trial times
 95: max_trial_times = 5;
 96:
 97: //D
 98: //V
 99: //inv_V
100: //inv_V * A * V
101:
102: // Solver
103: function [vec] = hat_g(t)
104:   vec = inv_V * g(t); // Faster!
105: //  vec = linsolve(V, -g(t)); // Slower!
```

```
106: //  vec = linsolve(l_V, -g(t));
107: //  vec = linsolve(u_V, vec);
108: endfunction
109:
110: hat_y0 = inv_V * y0;
111:
112: function g_i = gintvec_i(t, gfunc, i)
113:   vec = gfunc(t);
114:   g_i = vec(i);
115: endfunction
116:
117: //disp("integration of g");
118: //intg(0, 2, list(gintvec_i, g, 1))
119: //intg(0, 2, list(gintvec_i, g, 2))
120:
121: //lambda = spec(A);
122: //lambda(1)
123:
124: function ret_i = exp_lambda_t_tau_g_i(tau, t, lambda, gfunc, lambda
_i, gfunc_i)
125:   ret_i = exp(lambda(lambda_i) * (t - tau)) * gintvec_i(tau, gfunc,
 gfunc_i);
126: endfunction
127:
128: function ret_i_re = exp_lambda_t_tau_g_i_re(tau, t, lambda, gfunc,
lambda_i, gfunc_i)
129:   ret_i_re = real(exp_lambda_t_tau_g_i(tau, t, lambda, gfunc, lambd
a_i, gfunc_i));
130: endfunction
131:
132: function ret_i_im = exp_lambda_t_tau_g_i_im(tau, t, lambda, gfunc,
lambda_i, gfunc_i)
133:   ret_i_im = imag(exp_lambda_t_tau_g_i(tau, t, lambda, gfunc, lambd
a_i, gfunc_i));
134: endfunction
135:
136: function [v_i] = exp_lambda_v_i(t_start, t_end, lambda, gfunc, lamb
da_i)
```

```
137:   for i=1:1:lode_dim
138:     tmp_re = intg(t_start, t_end, list(exp_lambda_t_tau_g_i_re, t_e
nd, lambda, gfunc, lambda_i, i), intg_aerr, intg_rerr)
139:     tmp_im = intg(t_start, t_end, list(exp_lambda_t_tau_g_i_im, t_e
nd, lambda, gfunc, lambda_i, i), intg_aerr, intg_rerr)
140:     v_i(i) = tmp_re + tmp_im * %i;
141:   end
142: endfunction
143:
144: function [v_i] = exp_lambda_v_i_for_best(t_start, t_end, lambda, gf
unc)
145:   for i=1:1:lode_dim
146:     tmp_re = intg(t_start, t_end, list(exp_lambda_t_tau_g_i_re, t_e
nd, lambda, gfunc, i, i), intg_aerr, intg_rerr)
147:     tmp_im = intg(t_start, t_end, list(exp_lambda_t_tau_g_i_im, t_e
nd, lambda, gfunc, i, i), intg_aerr, intg_rerr)
148:     v_i(i) = tmp_re + tmp_im * %i;
149:   end
150: endfunction
151:
152: function [P_i] = exp_P_i(mat, lambda, lambda_i)
153:   P_i = eye(mat);
154:   for j=1:1:lambda_i-1
155:     P_i = P_i * (mat - lambda(j) * eye(mat)) / (lambda(lambda_i) -
lambda(j));
156:   end
157:   for j=lambda_i+1:1:lode_dim
158:     P_i = P_i * (mat - lambda(j) * eye(mat)) / (lambda(lambda_i) -
lambda(j));
159:   end
160: endfunction
161:
162: function [P_i_y] = exp_P_i_y(mat, lambda, lambda_i, y)
163:   P_i_y = y;
164:   for j = lode_dim:-1:lambda_i+1
165:     P_i_y = (mat - lambda(j) * eye(mat)) * P_i_y / (lambda(lambda_i
) - lambda(j));
166:   end
```

```
167:   for j = lambda_i-1:-1:1
168:     P_i_y = (mat - lambda(j) * eye(mat)) * P_i_y / (lambda(lambda_i
) - lambda(j));
169:   end
170: endfunction
171:
172: // Linear ODE solver (best case)
173: function [y] = best_lode_solver(mat, t_end, gfunc, init_y)
174:   lambda = spec(mat);
175:
176: //  printf("||A - V * lambda * inv_V|| / ||A||: %g\n", norm(mat - V
 * diag(lambda) * inv_V, 2) / norm(mat, 2));
177:
178:   init_y = inv_V * init_y;
179:
180:   for i = 1:1:lode_dim
181:     y(i) = exp(lambda(i) * t_end) * init_y(i);
182:   end
183:   y = y + exp_lambda_v_i_for_best(0, t_end, lambda, gfunc);
184: //  y = y + exp_lambda_v_i_for_best(0, t_end, lambda, hat_g);
185:   y = V * y;
186: endfunction
187:
188: printf("Best_y\n");
189: total_best_y_time = 0;
190: for trial_times = 1:1:max_trial_times
191:   tic();
192:   best_y = best_lode_solver(A, t_end, hat_g, y0);
193:   best_y_time = toc();
194:   printf("best_y(%d/%d): %f (sec)\n", trial_times, max_trial_times,
 best_y_time);
195:   total_best_y_time = total_best_y_time + best_y_time;
196: end
197:
198: printf("ODE solver(default)\n");
199: total_ode_default_y_time = 0;
200: for trial_times = 1:1:max_trial_times;
201:   tic();
```

```
202:   ode_default_y = ode(y0, 0, t_end, ode_rerr, ode_aerr, ode_func);
203:   ode_default_y_time = toc();
204:   printf("ode_y_default_time(%d/%d): %f\n", trial_times, max_trial_
times, ode_default_y_time);
205:   total_ode_default_y_time = total_ode_default_y_time + ode_default
_y_time;
206: end
207:
208: printf("ODE solver(stiff)\n");
209: total_ode_stiff_y_time = 0;
210: for trial_times = 1:1:max_trial_times;
211:   tic();
212:   ode_stiff_y = ode("stiff", y0, 0, t_end, ode_rerr, ode_aerr, ode_
func);
213:   ode_stiff_y_time = toc();
214:   printf("ode_y_stiff_time(%d/%d): %f\n", trial_times, max_trial_ti
mes, ode_stiff_y_time);
215:   total_ode_stiff_y_time = total_ode_stiff_y_time + ode_stiff_y_tim
e;
216: end
217:
218: printf("time(sec): %f, %f, %f\n", total_best_y_time/trial_times, to
tal_ode_default_y_time/trial_times, total_ode_stiff_y_time/trial_times);
219: printf("Relerr(best_y): %e, %e\n", relerr(ode_default_y, best_y), r
elerr(ode_stiff_y, best_y));
220: printf("y(1), y(n): %e, %e\n", best_y(1), best_y(lode_dim))
```