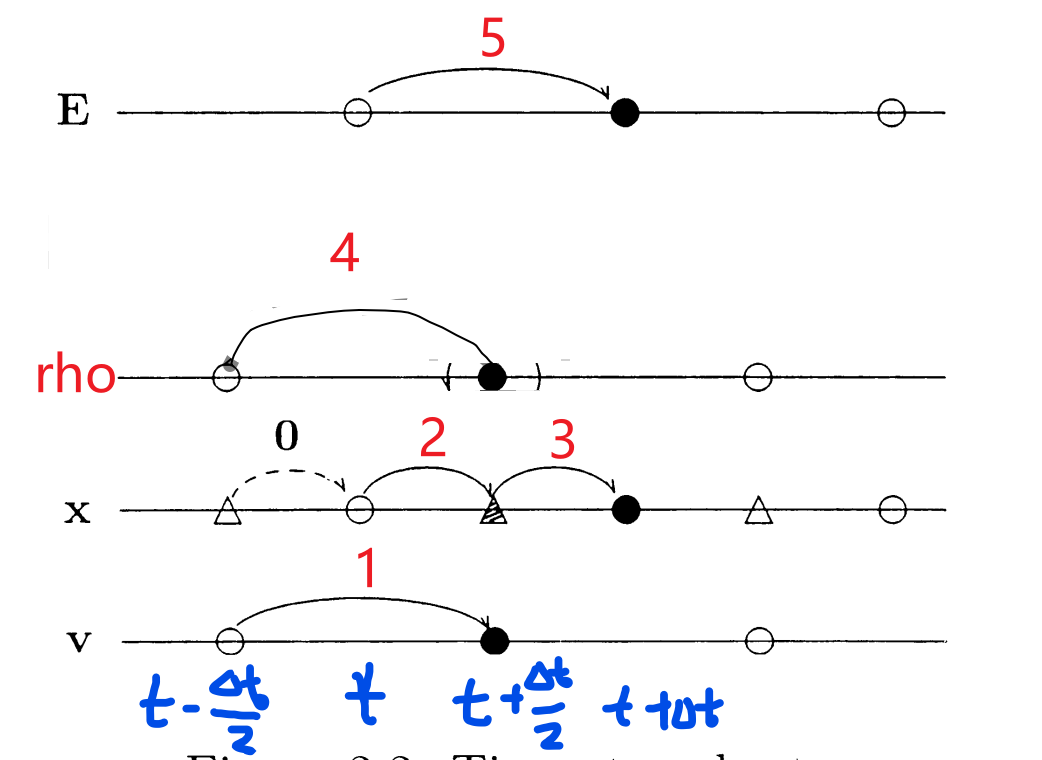
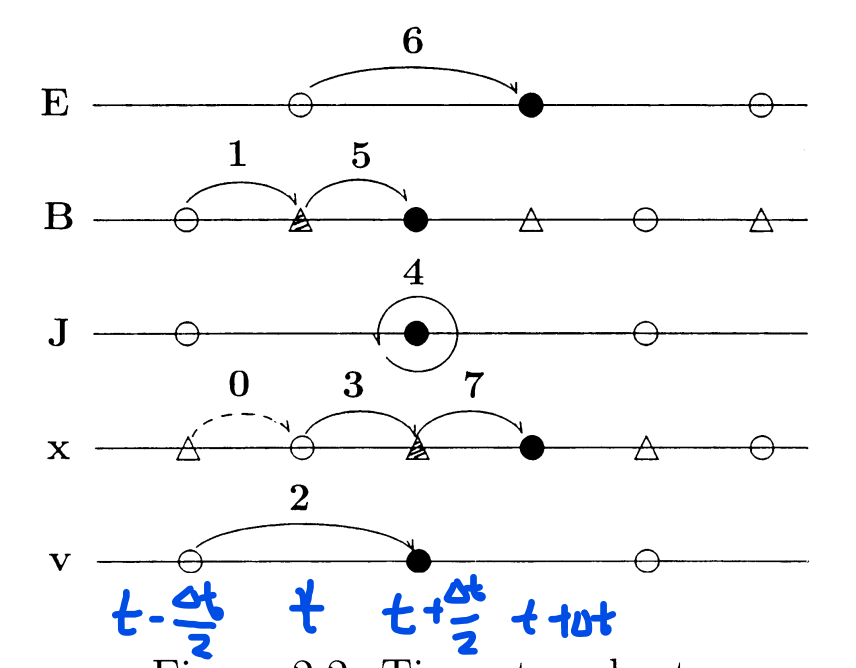
**%1 kempo1main.m**

1. function kempo1main
2. %\*\*\*\*\*\*\*\*\*\*\*\*\*伪随机数\*\*\*\*\*\*\*\*\*\*\*\*%
3. %用于随机生成粒子的初始坐标和相位角
4. rng('default');
5. rng(1);
6. global flag\_exit
7. flag\_exit=0
9. %\*\*\*\*\*\*\*\*\*\*\*\*\*读取参数\*\*\*\*\*\*\*\*\*\*\*\*%
10. prm = Parameters
12. %\*\*\*\*\*\*\*\*\*\*\*\*\*\*归一化\*\*\*\*\*\*\*\*\*\*\*\*\*%
13. [prm,ren] = renorm(prm);
15. %\*\*\*\*\*\*\*\*\*\*\*\*\*\*初始化\*\*\*\*\*\*\*\*\*\*\*\*\*%
16. %初始化画图设置，粒子坐标和速度等
17. [hdiag,output] = diagnostics\_init(prm);
18. particle = Particle(prm);
19. field = Field(prm);
21. %初次更新粒子位置和求电荷量、电场分布
22. position(particle,prm); %首次更新半个时间步的粒子位置
23. **if** prm.iex   %静电开关 ：iex=1为电磁，iex=2为静电
24. charge(particle, field, prm);  %求网格点电荷密度
25. poisson(field, prm);   %通过泊松方程求电场分布
26. end
28. %\*\*\*\*\*\*\*\*\*\*\*\*main loop\*\*\*\*\*\*\*\*\*\*\*%
29. jtime=0;
30. jdiag=1;  %诊断计数
32. %-- Diagnostics at initial time --
33. hdiag=diagnostics(hdiag,particle,field,output,prm,jtime,jdiag,ren);
34. **if** prm.nplot == 0  %nplot: number of output
35. **return**
36. end




42. % Time advance loop
43. **for** jtime = 1:prm.ntime  %？时间间隔dt和总时间步数的选择
44. **if** prm.iex==2  %iex=2静电；iex=1电磁
45. 
46. rvelocity(particle, field, prm);
47. position(particle, prm);
48. position(particle, prm);  %？两次位置变化？？ 每次更新半个时间步位置
49. charge(particle, field, prm);
50. poisson(field, prm);
51. **else**
53. %
54. bfield(field,prm);
55. rvelocity(particle, field, prm);  %Boris方法
56. position(particle, prm);
57. current(particle, field, jtime, prm);
58. bfield(field, prm);
59. efield(field, prm);
60. position(particle, prm);
61. end
63. %-- 时变诊断diagnostics --
64. **if** mod(jtime,prm.ifdiag)==0
65. jdiag = jdiag+1;
66. hdiag = diagnostics(hdiag, particle, field,output, prm, jtime, jdiag,ren);
67. end
68. **if** flag\_exit
69. **break**;
70. end
71. end
72. %-- diagnostics --
73. **if** ~flag\_exit
74. diagnostics\_last(hdiag, prm, jtime,output,ren);
75. end
76. end

**%2 Parameter.m**

1. %\*\*\*\*\*\*\*\*读取输入参数\*\*\*\*\*\*\*\*\*%
2. classdef Parameters < handle
4. properties
5. **%网格距和时间步长，需满足库朗条件 且小于徳拜长度**
6. dx **double** {mustBePositive}
7. dt **double** {mustBePositive}
8. **%网格点数**
9. nx **double** {mustBePositive}
10. **%时间步数**
11. ntime **double** {mustBeInteger,mustBePositive}
12. % number of outputs
13. nplot **double** {mustBeInteger, mustBePositive}
14. %光速，程序输入的光速为20 ？？
15. cv **double** {mustBePositive}
16. %粒子的回旋频率
17. wc **double** {mustBeReal}
18. %外部电流的振幅jz
19. ajamp **double** {mustBeNonnegative}
20. %电场振幅
21. eamp **double** {mustBePositive}
22. %maximum range **for** plotting the electric field
23. emax **double** {mustBeReal}
24. %磁场的振幅
25. bamp **double** {mustBePositive}
26. %maximum range **for** plotting the magnetic field
27. bmax **double** {mustBeReal}
28. % control parameter **for** electrostatic option
29. iex **double** {mustBeMember(iex,[0,1,2])}
30. % maximum range **for** plotting velocity
31. vmax **double** {mustBeReal}
32. %number of bins **for** deriving the particle distribution function
33. nv **double** {mustBeInteger}
34. % 外部电流的频率jz
35. wj **double** {mustBeNonnegative}
36. % 粒子种类的数量
37. ns **double** {mustBeInteger, mustBePositive}
38. % number of particles **for** species
39. np **double** {mustBeInteger, mustBePositive}
40. % plasma frequency of species
41. wp **double** {mustBePositive}
42. % charge-to-mass ratio of species
43. qm **double** {mustBeReal}
44. % parallel thermal velocity of species
45. vpa **double** {mustBePositive}
46. % perpendicular thermal velocity of species
47. vpe **double** {mustBePositive}
48. % drift velocity of species
49. vd **double** {mustBeNonnegative}
50. %漂移速度和水平方向的夹角，满足以下关系（=pch\*pi/180）
51. %
52. %
53. pch **double** {mustBeGreaterThanOrEqual(pch,0), mustBeLessThanOrEqual(pch,180)}
54. %颜色开关
55. icolor
56. %画图开关
57. iparam
58. %诊断方式
59. diagtype
60. angle  **double** {mustBeGreaterThanOrEqual(angle,0), mustBeLessThanOrEqual(angle,90)}
61. end
63. % Actually I am not sure **if** **this** should be calculated only once. It
64. % might be better.
65. % For example, in the initialization part, slx has been used several
66. % times inside the nested loop. q is used in function charge.
67. properties (Dependent)
68. slx
69. npt
70. nxp1
71. nxp2
72. X1
73. X2
74. X3
75. cs
76. tcs
77. q
78. mass
79. rho0
80. bx0
81. by0
82. ifdiag
83. end
85. methods
86. function obj = Parameters(fname)
87. % read input parameters and set values
89. **if** nargin==0
90. fname = 'input\_tmp.dat';  % **default** input filename
91. end
93. **try**
94. fid = fopen(fname);
95. C = textscan(fid,'%s%s','delimiter','=;','commentstyle','matlab');
96. [StrName,StrValue] = C{:};
97. fclose(fid);
98. **catch**
99. errordlg(sprintf('Can''t open input file: %s',fname),'Error')
100. end
102. **for** l=1:length(StrName)
103. value = eval(**char**(StrValue(l)));
104. prmname = strtrim(StrName{l});
105. **switch** prmname
106. **case** 'dx'
107. obj.dx = value;
108. **case** 'dt'
109. obj.dt = value;
110. **case** 'nx'
111. obj.nx = value;
112. **case** 'ntime'
113. obj.ntime = value;
114. **case** 'nplot'
115. obj.nplot = value;
116. **case** 'cv'
117. obj.cv = value;
118. **case** 'wc'
119. obj.wc = value;
120. **case** 'ajamp'
121. obj.ajamp = value;
122. **case** 'eamp'
123. obj.eamp = value;
124. **case** 'emax'
125. obj.emax = value;
126. **case** 'bamp'
127. obj.bamp = value;
128. **case** 'bmax'
129. obj.bmax = value;
130. **case** 'iex'
131. %                   obj.iex = value;
132. obj.iex = 2;
133. **case** 'vmax'
134. obj.vmax = value;
135. **case** 'nv'
136. obj.nv = value;
137. **case** 'wj'
138. obj.wj = value;
139. **case** 'ns'
140. obj.ns = value;
141. **case** 'np'
142. obj.np = value;
143. **case** 'wp'
144. obj.wp = value;
145. **case** 'qm'
146. obj.qm = value;
147. **case** 'vpa'
148. obj.vpa = value;
149. **case** 'vpe'
150. obj.vpe = value;
151. **case** 'vd'
152. obj.vd = value;
153. **case** 'pch'
154. obj.pch = value;
155. **case** 'icolor'
156. obj.icolor = value;
157. **case** 'iparam'
158. obj.iparam = value;
159. **case** 'diagtype'
160. %                   obj.diagtype = value;
161. obj.diagtype =  [1, 4, 5,  10, 11.000000,15, 25.000000, 30.000000];
162. **case** 'angle'
163. obj.angle = value;
164. otherwise
165. error('Plese check input parameter %s.',prmname)
166. end
167. end
168. end
170. % Get the dependent vars
172. %网格点数
173. function value = get.slx(obj)
174. value = obj.nx;
175. end
177. %总的网格点数
178. function value = get.npt(obj)
179. %总的粒子数量
180. value = sum(obj.np(1:obj.ns));
181. end
183. %网格点数+1
184. function value = get.nxp1(obj)
186. value = obj.nx+1;
187. end
189. %网格点数+2
190. function value = get.nxp2(obj)
191. value = obj.nx+2;
192. end
194. %1:128
195. function value = get.X1(obj)
196. value = 1:obj.nx;
197. end
199. %2:129
200. function value = get.X2(obj)
201. value = 2:(obj.nx+1);
202. end
204. %3:130
205. function value = get.X3(obj)
206. value = 3:(obj.nx+2);
207. end
209. %光速平方
210. function value = get.cs(obj)
211. value = obj.cv^2;
212. end
214. %光速平方的2倍
215. function value = get.tcs(obj)
216. value = 2\*obj.cs;
217. end
219. %电荷量
220. function value = get.q(obj)
221. value = obj.nx ./ obj.np(1:obj.ns) .\* (obj.wp(1:obj.ns).^2) ./ ...
222. obj.qm(1:obj.ns);
223. end
225. % This is a **case** where you have inter-dependency.
226. function value = get.mass(obj)
227. value = obj.q ./ obj.qm(1:obj.ns);
228. end
230. %电荷密度
231. function value = get.rho0(obj)
232. value = -sum(obj.q(1:obj.ns) .\* obj.np(1:obj.ns)) / obj.nx \*...
233. ones(obj.nxp2,1);
234. end
236. function value = get.bx0(obj)
237. theta = pi/180\*obj.angle;
238. value = obj.wc/obj.qm(1)\*cos(theta);
239. end
241. function value = get.by0(obj)
242. theta = pi/180\*obj.angle;
243. value = obj.wc/obj.qm(1)\*sin(theta);
244. end
246. %每次画的步长数
247. function value = get.ifdiag(obj)
248. value = ceil(obj.ntime/obj.nplot);
249. end
250. end
251. end

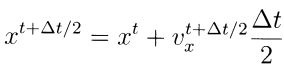
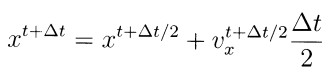
**%3 renorm.m**

1. %\*\*\*\*\*\*\*\*\*\*归一化系数\*\*\*\*\*\*\*\*\*%
2. %ren.\*=实际/模拟（归一化指的是网格距归一，其他参数并不归一而是等比例变化）
3. function [prm,ren]=renorm(prm)
4. ren.x=prm.dx             %网格距系数，将网格距归一化为1
5. ren.t=prm.dt/2           %时间步长系数，将时间步统一化为2
6. ren.v=ren.x/ren.t        %速度系数
7. ren.e=ren.x/(ren.t^2)    %电场系数
8. ren.b=1.0/ren.t          %磁场系数
9. ren.j=ren.x/(ren.t^3)    %电流密度系数
10. ren.r=1.0/(ren.t^2)      %电荷密度系数
11. ren.s=(ren.x^2)/(ren.t^4)%能量密度系数
13. prm.cv=prm.cv/ren.v      %等比列变化后的光速
14. prm.wc=prm.wc\*ren.t      %等比列变化后的回旋频率
16. prm.wp=prm.wp   .\*ren.t  **%等比列变化后的等离子体频率**
17. prm.vpa=prm.vpa ./ren.v  %等比列变化后的平行速度
18. prm.vpe=prm.vpe ./ren.v  %等比列变化后的垂直速度
19. prm.vd=prm.vd   ./ren.v  %等比列变化后的漂移速度
21. prm.vmax=prm.vmax ./ren.v%等比列变化后的速度上限
23. prm.wj=prm.wj\*ren.t      %等比列变化后的外部电流的频率jz
24. prm.ajamp=prm.ajamp/ren.j%等比列变化后的外部电流的振幅ajamp
26. end

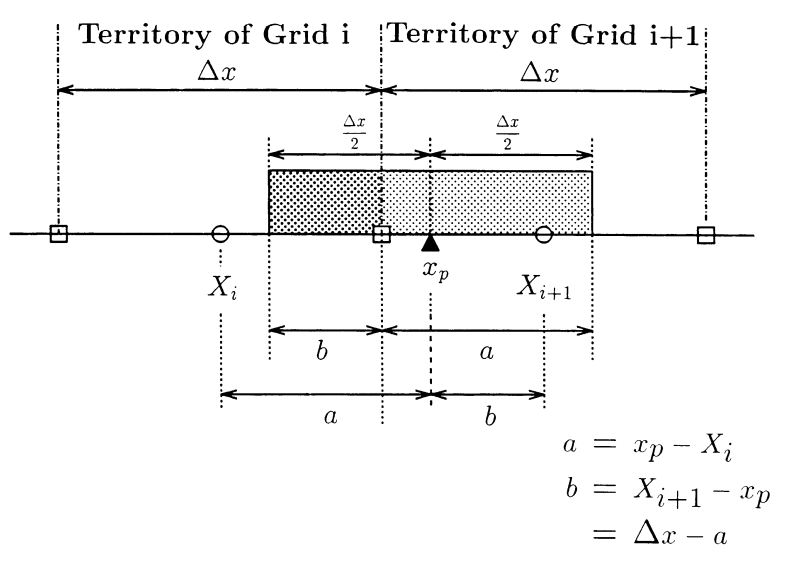
**%4 Particle.m**

1. classdef Particle < handle
3. properties
4. x **double**    %粒子坐标
5. vx **double**
6. vy **double**
7. vz **double**
8. end
10. methods
11. function obj=Particle(prm)
12. %粒子初始化
13. obj.x =zeros(prm.npt,1);
14. obj.vx=zeros(prm.npt,1);
15. obj.vy=zeros(prm.npt,1);
16. obj.vz=zeros(prm.npt,1);
18. n2=0;
19. **for** k=1:prm.ns
20. n1=n2;
21. n2=n2+prm.np(k);
23. phi = pi/180.0\*prm.pch(k); %漂移速度与水平方向的夹角
24. vdpa = prm.vd(k)\*cos(phi); %漂移速度的平行分量
25. vdpe = prm.vd(k)\*sin(phi); %漂移速度的垂直分量
27. xx = 0;
28. nphase = 1;
29. phase = 0;
31. %对每一个粒子进行操作
32. **for** i=(n1+1):n2
33. **if** mod(i,nphase) == 0
34. phase = 2\*pi\*rand;            %随机粒子的相位
35. xx = xx+ prm.nx/prm.np(k);    %等间距分配每一个粒子坐标
36. **else**
37. phase = phase + 2\*pi/nphase;  %随机粒子相位角
38. end
40. obj.x(i) = xx;                   %初始化粒子坐标
41. **if** obj.x(i) < 0.0
42. obj.x(i) = obj.x(i) + prm.slx;%slx网格点数=nx
43. end
44. **if** obj.x(i) >= prm.slx
45. obj.x(i) = obj.x(i) - prm.slx;%保证粒子坐标在网格内
46. end
48. uxi = prm.vpa(k)\*randn + vdpa;           %平行热速度+平行漂移速度
49. uyi = prm.vpe(k)\*randn + vdpe\*cos(phase);%垂直热速度的y分量+平行漂移速度的y分量
50. uz  = prm.vpe(k)\*randn + vdpe\*sin(phase);%垂直热速度的z分量+平行漂移速度的z分量
52. % rotation to the direction of the magnetic field
53. % angle为静磁场B0与波矢k的夹角，其中B0在x-y平面内
54. costh = cos(pi/180\*prm.angle); %costh=1
55. sinth = sin(pi/180\*prm.angle); %sinth=0
56. ux = costh\*uxi - sinth\*uyi;    %当angle=0时，
57. uy = sinth\*uxi + costh\*uyi;    %静磁场对速度没偏转
59. %？抑制初始场的波动(Birdsall and Langdon[1985])
60. g = prm.cv /sqrt(prm.cs + ux\*ux + uy\*uy + uz\*uz);
61. obj.vx(i) = ux\*g;
62. obj.vy(i) = uy\*g;
63. obj.vz(i) = uz\*g;
64. end
65. end
66. end
67. end
68. end

**%5 position.m**

1. function particle = position(particle,prm)
2. % Update the position in one step
4. slx = prm.slx
5. p = particle
7. %更新半个时间步的位置，时间内需调用2次(注：一个时间步为2)
8. %
9. %
10. p.x = p.x + p.vx
11. % 周期性边界条件：保证粒子在网格内  Periodic BC
12. p.x = p.x + slx.\*(p.x<0.0) - slx.\*(p.x>=slx)
13. end

**%6 charge.m**

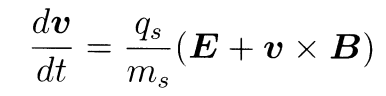
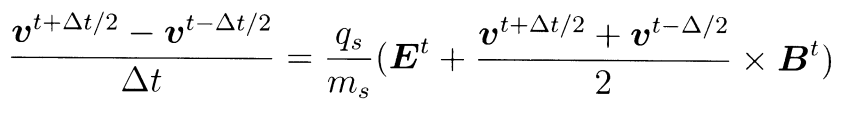
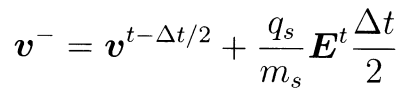
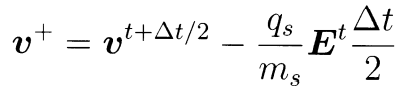
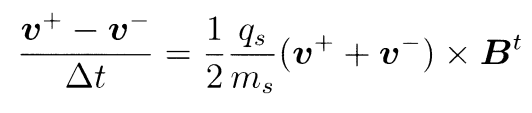
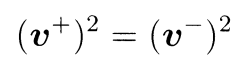
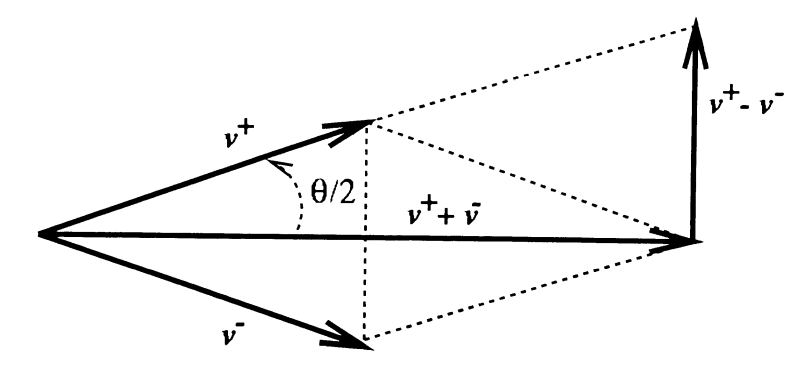
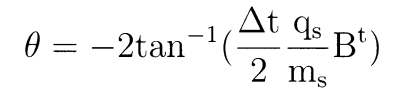
1. function charge(particle,field,prm)
2. % Calculate the charge on the grid from particles
4. p = particle; % reference to the particle obj
6. % Field.rho = zeros(prm.nxp2,1);
7. field.rho = prm.rho0; %  ->
9. n2 = 0;
10. **for** k=1:prm.ns
11. n1 = n2;
12. n2 = n1 + prm.np(k);
13. **for** m = (n1+1):n2
14. i  = floor(p.x(m) + 2.0);           %每个粒子所在的网格点坐标
15. i1 = i + 1;                         %每个粒子坐标的下一个网格点坐标
16. s2 = (p.x(m)+ 2.0 - i)\*prm.q(k);    %分到右侧网格点的电荷密度
17. s1 = prm.q(k) - s2;                 %分配到左侧网格点的电荷量
18. 
19. field.rho(i ) = field.rho(i ) + s1; %累加分配到左侧网格的电荷量
20. field.rho(i1) = field.rho(i1) + s2; %累加分配到右侧网格的电荷量
21. %？这里分配的电荷密度是否归一化 %  ->
22. end
23. end
25. %电荷密度边界条件
26. field.rho(2) = field.rho(2) + field.rho(prm.nxp2) - prm.rho0(2);%?
27. field.rho(1) = field.rho(prm.nxp1);
28. field.rho(prm.nxp2) = field.rho(2);

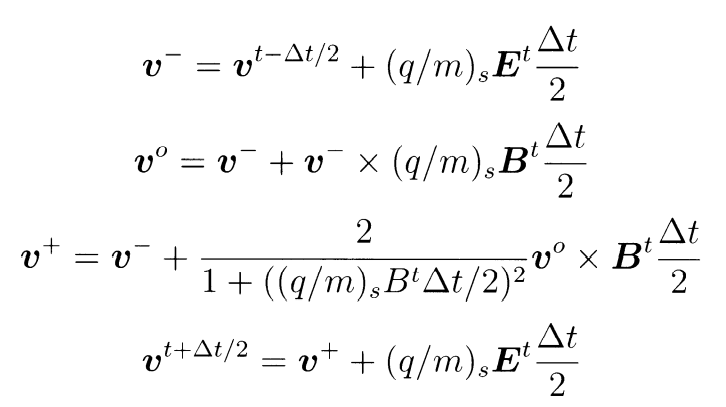
31. end

**%7 poisson.m**

1. function poisson(field, prm)
2. % Calculate Ex from Poisson equation
4. f = field; % reference to the Field obj
6. %泊松方程
7. %差分形式
8. %这里定义真空介电常数等于1
9. % ,
10. f.ex(prm.X2) = f.ex(prm.X2-1) + f.rho(prm.X2);
12. ex0 = sum(f.ex(prm.X2))/prm.nx      %所有网格点的平均电场强度
13. f.ex(prm.X2) = f.ex(prm.X2) - ex0;  %？？？
14. f.ex(1) = f.ex(prm.nxp1);           %对称边界条件 nxp1=nx+1
15. f.ex(prm.nxp2) = f.ex(2);           %对称边界条件 nxp2=nx+2
16. end

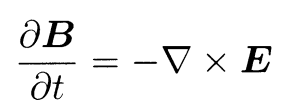
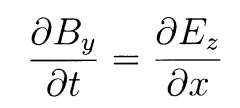
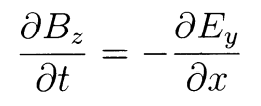
**%8 rvelocity.m**

1. function rvelocity(particle, field, prm)
2. % Update velocity in one step
4. % References to **class** obj
5. p = particle;
6. f = field;
8. nxp1 = prm.nxp1; nxp2 = prm.nxp2; ns = prm.ns; np = prm.np;
9. qm = prm.qm;
10. bx0 = prm.bx0;  %bx0=prm.wc/prm.qm(1)\*cos(theta);
11. cs = prm.cs;
12. X1 = prm.X1; X2 = prm.X2; X3 = prm.X3; %X2=2:(prm.nx+1);X3=3:(prm.nx+2)
14. %\*\*\*\*\*\*\*\*通过Boris方法更新粒子速度\*\*\*\*\*\*\*%
15. %牛顿第二定律
16. %差分形式
17. %定义
18. %定义
19. %将和代入差分形式得
20. %两边同乘以得
21. %Boris方法矢量关系图
22. %由矢量可以求得夹角，即

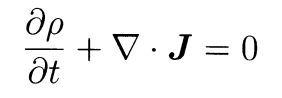
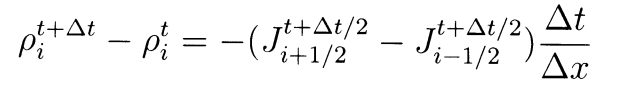
%

1. work1 = zeros(nxp2,1);
2. work2 = zeros(nxp2,1);
3. work1(X2) = 0.5\*(f.ex(X1) + f.ex(X2));
4. work1(nxp2)= work1(2);
5. work2(X2) = 0.5\*(f.by(X3) + f.by(X2));
6. work2(1) = work2(nxp1);
8. %-----
9. n2 = 0;
10. **for** k=1:ns
11. n1 = n2;
12. n2 = n2 + np(k);
14. **for** m=(n1+1):n2
15. i = floor(p.x(m) + 2.0);
16. sf2 = (p.x(m) + 2.0 - i)\*qm(k);
17. sf1 = qm(k) - sf2;
19. ih = floor(p.x(m) + 1.5);
20. sh2 = (p.x(m) + 1.5 - ih)\*qm(k);
21. sh1 = qm(k) - sh2;
23. i1 = i + 1;
24. ih1 = ih + 1;
26. ex1 = sf1\*work1(i) + sf2\*work1(i1);
27. ey1 = sf1\*f.ey(i)  + sf2\*f.ey( i1);
28. ez1 = sh1\*f.ez(ih) + sh2\*f.ez(ih1);
30. bx1 = bx0\*prm.qm(k);
31. by1 = sh1\*work2(ih) + sh2\*work2(ih1);
32. bz1 = sh1\*f.bz(ih)  + sh2\*f.bz(ih1);
34. g = prm.cv / sqrt(cs - p.vx(m)^2 - p.vy(m)^2 - p.vz(m)^2);
36. ux = p.vx(m)\*g + ex1;%？
37. uy = p.vy(m)\*g + ey1;
38. uz = p.vz(m)\*g + ez1;
40. g = prm.cv/sqrt(cs + ux\*ux + uy\*uy + uz\*uz);
42. bx1 = bx1\*g;
43. by1 = by1\*g;
44. bz1 = bz1\*g;
46. boris = 2.0/(1+ bx1\*bx1 + by1\*by1 + bz1\*bz1);
48. uxt = ux + uy\*bz1 - uz\*by1;
49. uyt = uy + uz\*bx1 - ux\*bz1;
50. uzt = uz + ux\*by1 - uy\*bx1;
52. ux = ux + boris\*(uyt\*bz1 - uzt\*by1) + ex1;
53. uy = uy + boris\*(uzt\*bx1 - uxt\*bz1) + ey1;
54. uz = uz + boris\*(uxt\*by1 - uyt\*bx1) + ez1;
56. g = prm.cv /sqrt(cs + ux\*ux + uy\*uy + uz\*uz);
58. % **this** is causing speed issue. Why?
59. p.vx(m) = ux\*g;
60. p.vy(m) = uy\*g;
61. p.vz(m) = uz\*g;
62. end
63. end
65. end

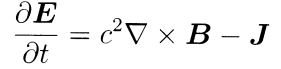
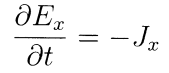
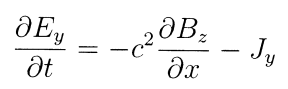
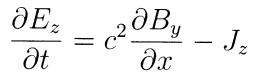
**%9 bfield.m**

1. function [field] = bfield(field,prm)
2. % Update magnetic field in one step
3. X2 = prm.X2;  %2:129
4. f = field; % reference to Field obj
6. %，标量形式如下：
7. %
8. % 
9. f.by(X2) = f.by(X2) + f.ez(X2) - f.ez(X2-1);
10. f.bz(X2) = f.bz(X2) - f.ey(X2+1) + f.ey(X2  );  %？
12. f.by(prm.nxp2)= f.by(2);%对称边界条件
13. f.bz(prm.nxp2)= f.bz(2);
14. f.by(1)   = f.by(prm.nxp1);
15. f.bz(1)   = f.bz(prm.nxp1);
16. end

**%10 current.m**

1. function current(particle, field, jtime, prm)
2. % Calculate the current in one step
4. nx = prm.nx; nxp1 = prm.nxp1; nxp2 = prm.nxp2;
5. X2 = prm.X2;
6. np = prm.np; ns = prm.ns;
7. q = prm.q;
9. % references to **class** obj
10. f = field;
11. p = particle;
13. f.ajx = zeros(nxp2,1);
14. f.ajy = zeros(nxp2,1);
15. f.ajz = zeros(nxp2,1);
17. %----
18. n2 = 0;
19. **for** k=1:ns
20. n1 = n2;
21. n2 = n2 + np(k);
22. qh = q(k)\*0.5;
24. **for** m=(n1+1):n2
25. ih = floor( p.x(m) + 1.5 );
26. s2 = (p.x(m) + 1.5 - ih)\*q(k);
27. s1 = q(k) - s2;
28. ih1= ih+1;
29. %jy和jz按线性比重分配到相邻网格点
30. f.ajy(ih)  = f.ajy(ih)  + p.vy(m)\*s1;
31. f.ajy(ih1) = f.ajy(ih1) + p.vy(m)\*s2;
32. f.ajz(ih)  = f.ajz(ih)  + p.vz(m)\*s1;
33. f.ajz(ih1) = f.ajz(ih1) + p.vz(m)\*s2;
35. %--Jx采用电荷守恒方法—
36. %
37. %
38. **if** prm.iex
39. qhs = qh \* sign(p.vx(m));
40. avx = abs(p.vx(m));
42. x1 = p.x(m) + 2.0 - avx;
43. x2 = p.x(m) + 2.0 + avx;
44. i1 = floor(x1);
45. i2 = floor(x2);
47. % This is causing me speed issue
48. f.ajx(i1) = f.ajx(i1) + (i2 - x1)\*qhs;
49. f.ajx(i2) = f.ajx(i2) + (x2 - i2)\*qhs;
50. end
51. end
52. end
54. %-- boundary --
55. f.ajx(nxp1) = f.ajx(1) + f.ajx(nxp1);
56. f.ajx(2)    = f.ajx(2) + f.ajx(nxp2);
58. f.ajy(nxp1) = f.ajy(1) + f.ajy(nxp1);
59. f.ajy(2)    = f.ajy(2) + f.ajy(nxp2);
60. f.ajy(1)    = f.ajy(nxp1);
62. f.ajz(nxp1) = f.ajz(1) + f.ajz(nxp1);
63. f.ajz(2)    = f.ajz(2) + f.ajz(nxp2);
65. %--
66. f.ajy(X2) = 0.5\*(f.ajy(X2) + f.ajy(X2-1));
68. %-- external current source ----
69. **if** prm.ajamp
70. f.ajz(nx/2+1) = f.ajz(nx/2+1) + prm.ajamp\*sin(prm.wj\*jtime);
71. **else**
72. %--cancellation of uniform Jx,Jy,Jz components---
73. ajxu = sum(f.ajx(X2))/nx;
74. ajyu = sum(f.ajy(X2))/nx;
75. ajzu = sum(f.ajz(X2))/nx;
76. f.ajx(X2) = f.ajx(X2) - ajxu;
77. f.ajy(X2) = f.ajy(X2) - ajyu;
78. f.ajz(X2) = f.ajz(X2) - ajzu;
79. end
81. end

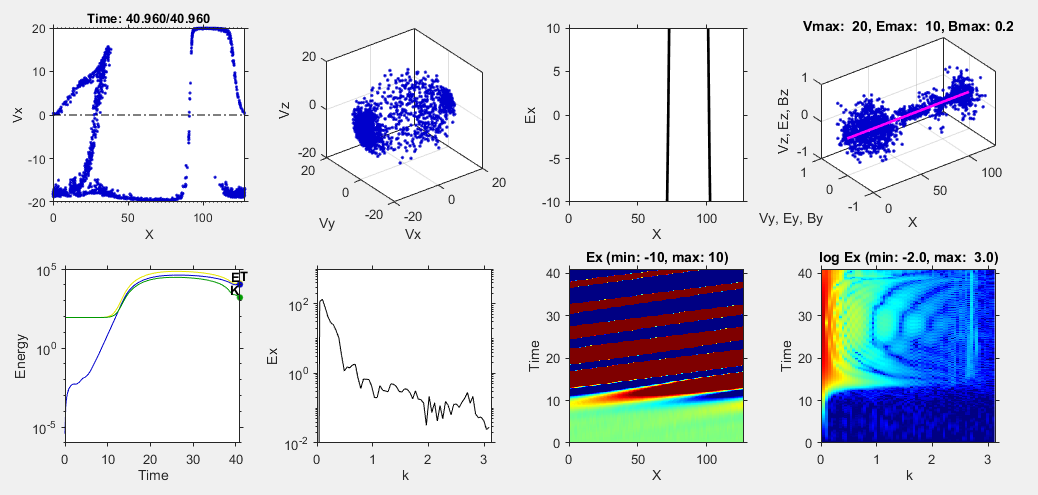
**%11 efield.m**

1. function [field] = efield(field,prm)
2. % Update the electric field in one step
4. nxp1 = prm.nxp1; nxp2 = prm.nxp2;
5. tcs = prm.tcs;                %2\*obj.cs;
6. X1 = prm.X1; X2 = prm.X2; X3 = prm.X3;
8. f = field; % reference to **class** obj
10. % % ,
11. %,标量形式如下
12. %
13. %
14. % 
15. **if** prm.iex == 0  %iex==0:没有电场
16. f.ex(:) = 0;
17. **else**
18. f.ex(X2) = f.ex(X2) - 2\*f.ajx(X2);
19. f.ex(1) = f.ex(nxp1);
20. f.ex(nxp2) = f.ex(2);
21. end
23. f.ey(X2) = f.ey(X2) - tcs\*(f.bz(X2) - f.bz(X1)) - 2\*f.ajy(X2);
24. f.ez(X2) = f.ez(X2) + tcs\*(f.by(X3) - f.by(X2)) - 2\*f.ajz(X2);
26. f.ey(1) = f.ey(nxp1);
27. f.ez(1) = f.ez(nxp1);
28. f.ey(nxp2) = f.ey(2);
29. f.ez(nxp2) = f.ez(2);
31. end

**%12 Inputs and Outputs**

%12.1 Inputs

1. dx = 1.000000;
2. dt = 0.040000;
3. nx = 128.000000;
4. ntime = 1024.000000;
5. cv = 20.000000;
6. wc = 0.000000;
7. angle = 0.000000;
9. ns = 1.000000;
10. np = [2048.000000, ];
11. wp = [1.000000, ];
12. qm = [-1.000000, ];
13. vpa = [0.500000, ];
14. vpe = [10.000000, ];
15. vd = [0.000000, ];
16. pch = [0.000000, ];
18. iex = 2;
20. ajamp = 0.000000;
21. wj = 0.000000;
23. nplot = 256.000000;
24. nv = 100.000000;
25. icolor = 1.000000;
26. iparam = 1.000000;
27. vmax = 20.000000;
28. emax = 10.000000;
29. bmax = 0.200000;
30. diagtype = [11.000000, 23.000000, 18.000000, 4.000000, ];

%12.2 Outputs(iex=2)

%12.3 Outputs(iex=1)

