
Plasma Program

UCLA Parallel PIC Framework, Viktor K. Decyk and Charles D. Norton in Comp. Phys. Communications 164 (2004) 80-85

PIC: parallel Particle-in-Cell codes, original idea is based on object-oriented in Fortran95.

■ serial one

```
In[10]= nx = 32; ny = 32; np = 10 240; nb = 2048; charge = 1; dt = 0.1
```

```
Out[10]= 0.1
```

```
In[11]= ChargeDeposit1[part_, charge_] := Module[{rho, i, j, x, ix, fx},
  rho = Table[0, {nx + 1}, {ny + 1}];
  Do[x = part[[i]][[1]]; ix = Floor[x]; fx = x - ix; rho[[ix[[1]] + 1, ix[[2]] + 1] =
    rho[[ix[[1]] + 1, ix[[2]] + 1] + (1 - fx[[1]]) * (1 - fx[[2]]) * charge;
    rho[[ix[[1]] + 2, ix[[2]] + 1] = rho[[ix[[1]] + 2, ix[[2]] + 1] +
      (fx[[1]]) * (1 - fx[[2]]) * charge; rho[[ix[[1]] + 1, ix[[2]] + 2] =
      rho[[ix[[1]] + 1, ix[[2]] + 2] + (1 - fx[[1]]) * (fx[[2]]) * charge; rho[[ix[[1]] + 2,
        ix[[2]] + 2] = rho[[ix[[1]] + 2, ix[[2]] + 2] + (fx[[1]]) * (fx[[2]]) * charge,
      {i, Length[part]};
  Do[rho[[1, i]] = rho[[1, i]] + rho[[1 + nx, i]], {i, ny};
  Do[rho[[i, 1]] = rho[[i, 1]] + rho[[i, 1 + ny]], {i, nx};
  Table[rho[[i, j]], {i, nx}, {j, ny}];

  KEDeposit1[part_] := Module[{rho, i, j, x, ix, fx, charge},
    rho = Table[0, {nx + 1}, {ny + 1}];
    Do[x = part[[i]][[1]]; charge = part[[i]][[2]][[1]]^2 + part[[i]][[2]][[2]]^2;
      ix = Floor[x]; fx = x - ix; rho[[ix[[1]] + 1, ix[[2]] + 1] =
        rho[[ix[[1]] + 1, ix[[2]] + 1] + (1 - fx[[1]]) * (1 - fx[[2]]) * charge;
        rho[[ix[[1]] + 2, ix[[2]] + 1] = rho[[ix[[1]] + 2, ix[[2]] + 1] +
          (fx[[1]]) * (1 - fx[[2]]) * charge; rho[[ix[[1]] + 1, ix[[2]] + 2] =
          rho[[ix[[1]] + 1, ix[[2]] + 2] + (1 - fx[[1]]) * (fx[[2]]) * charge; rho[[ix[[1]] + 2,
            ix[[2]] + 2] = rho[[ix[[1]] + 2, ix[[2]] + 2] + (fx[[1]]) * (fx[[2]]) * charge,
          {i, Length[part]};
    Do[rho[[1, i]] = rho[[1, i]] + rho[[1 + nx, i]], {i, ny};
    Do[rho[[i, 1]] = rho[[i, 1]] + rho[[i, 1 + ny]], {i, nx};
    Table[rho[[i, j]], {i, nx}, {j, ny}];

  FieldSolve1[rho_, kernel_] := Module[{rhotilde, i, j},
    rhotilde = Fourier[rho]; InverseFourier[Table[rhotilde[[i, j]] * kernel[[i, j]],
      {i, Length[rhotilde]}, {j, Length[rhotilde[[1]]}]];

  FieldStitch1[fx_, fy_] :=
    Table[{fx[[i, j]], fy[[i, j]]}, {i, Length[fx]}, {j, Length[fx[[1]]}];

  ParticlePush1[part_, inField_] := Module[{ix, fx, ax, nv, i, j, field},
    field = Table[inField[[1 + Mod[i, nx], 1 + Mod[j, ny]]], {i, nx + 1}, {j, ny + 1}]; Table[
      ix = Floor[part[[i]][[1]];
      fx = part[[i]][[1]] - ix; ax = (field[[ix[[1]] + 1, ix[[2]] + 1] * (1 - fx[[1]]) +
        fx[[1]] * field[[ix[[1]] + 2, ix[[2]] + 1]) * (1 - fx[[2]]) +
```

```

    fx[[2]] * (field[[ix[[1]] + 1, ix[[2]] + 2]] * (1 - fx[[1]]) +
    fx[[1]] * field[[ix[[1]] + 2, ix[[2]] + 2]]);
    nv = part[[i]][[2]] + ax * dt; {part[[i]][[1]] + nv * dt, nv}, {i, Length[part]}}];

ParticleManagel[part_] := Module[{i, x, list},
  list = Table[
    x = part[[i]][[1]]; x = x + {If[x[[1]] < 0, +nx, If[x[[1]] ≥ nx, -nx, 0]}, If[x[[2]] < 0,
    +ny, If[x[[2]] ≥ ny, -ny, 0]}}; {x, part[[i]][[2]]}, {i, Length[part]}}];

DrawThreeel[speed_, rho_, potential_] := Module[{r, g, b},
  r = Transpose[MyNormalize[speed]];
  g = Transpose[MyNormalize[rho]];
  b = Transpose[MyNormalize[potential]]; If[True, Show[Graphics[
  Raster[Table[List[PixelDouble[r, i, j], PixelDouble[g, i, j], PixelDouble[b, i, j]],
  {i, 2 * Length[r] - 1}, {j, 2 * Length[r[[1]]] - 1}], AspectRatio → Automatic]]];

MyNormalize[list_] := Module[{i, j, mx, mn},
  mx = mn = list[[1, 1]];
  Do[If[list[[i, j]] > mx, mx = list[[i, j]]; If[list[[i, j]] < mn, mn = list[[i, j]]],
  {i, Length[list]}, {j, Length[list[[1]]]}; If[mx > mn,
  Table[(list[[i, j]] - mn) / (mx - mn), {i, Length[list]}, {j, Length[list]}, 0]];

PixelDouble[l_, i_, j_] := If[Mod[j, 2] < 1,
  If[Mod[i, 2] < 1, (1[[IntegerPart[(i - 1) / 2] + 1, IntegerPart[(j - 1) / 2] + 1]] +
  1[[1 + IntegerPart[(i - 1) / 2], 2 + IntegerPart[(j - 1) / 2]]] +
  1[[2 + IntegerPart[(i - 1) / 2], 1 + IntegerPart[(j - 1) / 2]]] +
  1[[2 + IntegerPart[(i - 1) / 2], 2 + IntegerPart[(j - 1) / 2]])] / 4,
  (* else *) (1[[IntegerPart[(i - 1) / 2] + 1, IntegerPart[(j - 1) / 2] + 2]] +
  1[[IntegerPart[(i - 1) / 2] + 1, IntegerPart[(j - 1) / 2] + 1]]) / 2,
  (* else *) If[Mod[i, 2] < 1, (1[[IntegerPart[(i - 1) / 2] + 1, IntegerPart[(j - 1) / 2] + 1]] +
  1[[2 + IntegerPart[(i - 1) / 2], 1 + IntegerPart[(j - 1) / 2]])] / 2,
  (* else *)
  1[[IntegerPart[(i - 1) / 2] + 1, IntegerPart[(j - 1) / 2] + 1]]];

potkernel = Table[If[i == 1 && j == 1, I,
  If[j * 2 > ny, If[i * 2 > nx, 1 / ((nx + 1 - i) * (nx + 1 - i) + (ny + 1 - j) * (ny + 1 - j)),
  1 / ((i - 1) * (i - 1) + (ny + 1 - j) * (ny + 1 - j))],
  If[i * 2 > nx, 1 / ((nx + 1 - i) * (nx + 1 - i) + (j - 1) * (j - 1)),
  1 / ((i - 1) * (i - 1) + (j - 1) * (j - 1))]] + I], {i, nx}, {j, ny}}];

fieldXkernel = Table[If[i == 1 && j == 1, 0, -I *
  If[j * 2 > ny, If[i * 2 > nx, (nx + 1 - i) / ((nx + 1 - i) * (nx + 1 - i) + (ny + 1 - j) * (ny + 1 - j)),
  (i - 1) / ((i - 1) * (i - 1) + (ny + 1 - j) * (ny + 1 - j))],
  If[i * 2 > nx, (nx + 1 - i) / ((nx + 1 - i) * (nx + 1 - i) + (j - 1) * (j - 1)),
  (i - 1) / ((i - 1) * (i - 1) + (j - 1) * (j - 1))]]], {i, nx}, {j, ny}}];

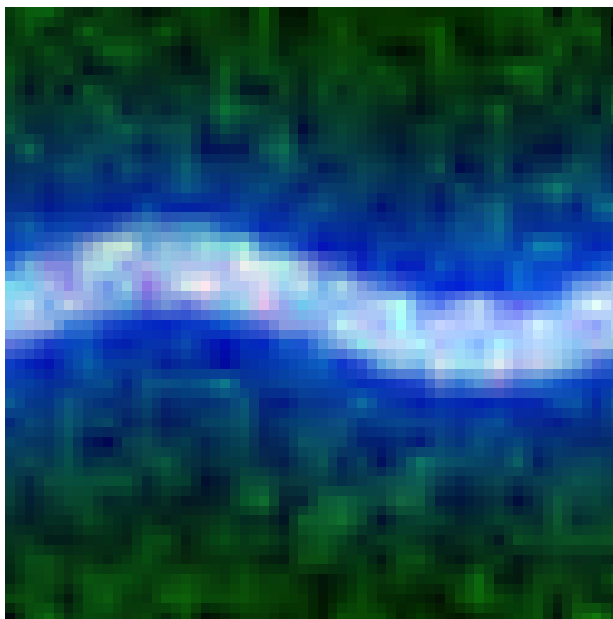
fieldYkernel = Table[If[i == 1 && j == 1, 0, -I *
  If[j * 2 > ny, If[i * 2 > nx, (ny + 1 - j) / ((nx + 1 - i) * (nx + 1 - i) + (ny + 1 - j) * (ny + 1 - j)),
  (ny + 1 - j) / ((i - 1) * (i - 1) + (ny + 1 - j) * (ny + 1 - j))],
  If[i * 2 > nx, (j - 1) / ((nx + 1 - i) * (nx + 1 - i) + (j - 1) * (j - 1)),
  (j - 1) / ((i - 1) * (i - 1) + (j - 1) * (j - 1))]]], {i, nx}, {j, ny}}];

In[12]= particles =
  If[True, Table[If[i > np, x = Random[]; {{x * nx, (ny / 2) + (Sin[x * Pi * 2] * ny / 16) +
  ((Random[] - 0.5) * ny / 8)}, {(Random[]) * 0.5, 0.5 * (Random[] - 0.5)}},
  {{Random[] * nx, Random[] * ny}, {(Random[] - 0.5) / 4, (Random[] - 0.5) / 4}}],
  {i, np + nb}], List[]]; particles = ParticleManagel[particles];

```

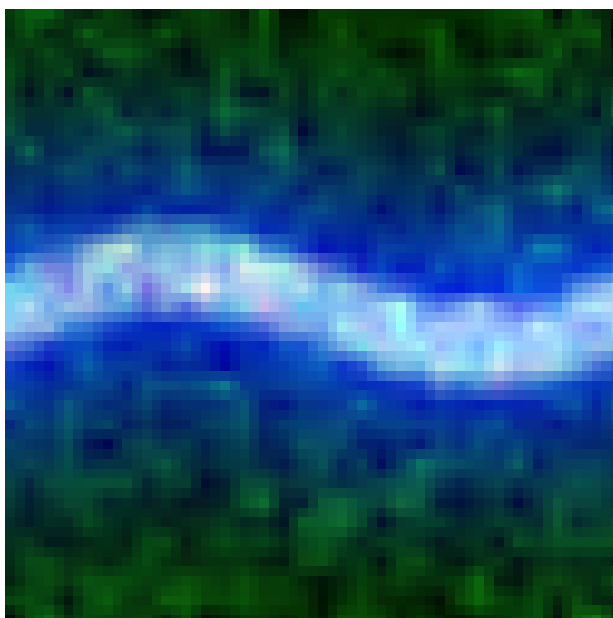
```
In[13]= {Timing[rho = ChargeDeposit1[particles, charge]; field =
  FieldStitch1[Re[FieldSolve1[rho, fieldXkernel]], Re[FieldSolve1[rho, fieldYkernel]]];
  potential = Re[FieldSolve1[rho, potkernel]];
  particles = ParticleManagel[ParticlePush1[particles, Re[field]]];
  ke = KEDeposit1[particles];], DrawThree1[ke, rho, potential]}
```

```
Out[13]= {{2.96845, Null},
```



```
In[14]= {Timing[Do[rho = ChargeDeposit1[particles, charge]; field = FieldStitch1[
  Re[FieldSolve1[rho, fieldXkernel]], Re[FieldSolve1[rho, fieldYkernel]]];
  particles = ParticleManagel[ParticlePush1[particles, Re[field]], {4}];],
  ke = KEDeposit1[particles]; potential = Re[FieldSolve1[rho, potkernel]];
  DrawThree1[ke, rho, potential]}
```

```
Out[14]= {{8.84233, Null},
```



■ Parallel version

```
In[15]= nx = 32; ny = 32; np = 10 240; nb = 2048; charge = 1; dt = 0.1
```

```
Out[15]= 0.1
```

```

In[16]:= idproc = mpiCommRank[mpiCommWorld];
nproc = mpiCommSize[mpiCommWorld]; mpiGather[{idproc, nproc}]

Out[16]= {{0, 8}, {1, 8}, {2, 8}, {3, 8}, {4, 8}, {5, 8}, {6, 8}, {7, 8}}

In[17]:= ChargeDeposit[part_, charge_] := Module[{rho, i, j, x, ix, fx},
  rho = Table[0, {nx + 1}, {ny + 1}];
  Do[x = part[[i]][[1]]; ix = Floor[x]; fx = x - ix; rho[[ix[[1]] + 1, ix[[2]] + 1] =
    rho[[ix[[1]] + 1, ix[[2]] + 1] + (1 - fx[[1]]) * (1 - fx[[2]]) * charge;
    rho[[ix[[1]] + 2, ix[[2]] + 1] = rho[[ix[[1]] + 2, ix[[2]] + 1] +
    (fx[[1]]) * (1 - fx[[2]]) * charge; rho[[ix[[1]] + 1, ix[[2]] + 2] =
    rho[[ix[[1]] + 1, ix[[2]] + 2] + (1 - fx[[1]]) * (fx[[2]]) * charge;
    rho[[ix[[1]] + 2, ix[[2]] + 2] = rho[[ix[[1]] + 2, ix[[2]] + 2] +
    (fx[[1]]) * (fx[[2]]) * charge, {i, Length[part]};
  Do[rho[[1, i]] = rho[[1, i]] + rho[[1 + nx, i]], {i, ny};
  Do[rho[[i, 1]] = rho[[i, 1]] + rho[[i, 1 + ny]], {i, nx};
  Table[rho[[i, j]], {i, nx}, {j, ny}];

  KEDeposit[part_] := Module[{rho, i, j, x, ix, fx, charge},
    rho = Table[0, {nx + 1}, {ny + 1}];
    Do[x = part[[i]][[1]];
      charge = part[[i]][[2]][[1]]^2 + part[[i]][[2]][[2]]^2; ix = Floor[x];
      fx = x - ix; rho[[ix[[1]] + 1, ix[[2]] + 1] = rho[[ix[[1]] + 1, ix[[2]] + 1] +
      (1 - fx[[1]]) * (1 - fx[[2]]) * charge; rho[[ix[[1]] + 2, ix[[2]] + 1] =
      rho[[ix[[1]] + 2, ix[[2]] + 1] + (fx[[1]]) * (1 - fx[[2]]) * charge;
      rho[[ix[[1]] + 1, ix[[2]] + 2] = rho[[ix[[1]] + 1, ix[[2]] + 2] +
      (1 - fx[[1]]) * (fx[[2]]) * charge; rho[[ix[[1]] + 2, ix[[2]] + 2] =
      rho[[ix[[1]] + 2, ix[[2]] + 2] + (fx[[1]]) * (fx[[2]]) * charge, {i, Length[part]};
    Do[rho[[1, i]] = rho[[1, i]] + rho[[1 + nx, i]], {i, ny};
    Do[rho[[i, 1]] = rho[[i, 1]] + rho[[i, 1 + ny]], {i, nx};
    Table[rho[[i, j]], {i, nx}, {j, ny}];

  FieldSolve[inRho_, kernel_] := Module[{rho, rhotilde, i, j},
    mpiAllreduce[inRho, rho, mpiSum, mpiCommWorld];
    rhotilde = Fourier[rho]; InverseFourier[Table[rhotilde[[i, j]] * kernel[[i, j]],
      {i, Length[rhotilde]}, {j, Length[rhotilde[[1]]}]];

  FieldStitch[fX_, fY_] :=
    Table[{fX[[i, j]], fY[[i, j]]}, {i, Length[fX]}, {j, Length[fX[[1]]}];

  ParticlePush[part_, inField_] := Module[{ix, fx, ax, nv, i, j, field},
    field = Table[inField[[1 + Mod[i, nx], 1 + Mod[j, ny]]], {i, nx + 1}, {j, ny + 1}];
    Table[
      ix = Floor[part[[i]][[1]]]; fx = part[[i]][[1]] - ix;
      ax = (field[[ix[[1]] + 1, ix[[2]] + 1] * (1 - fx[[1]]) +
        fx[[1]] * field[[ix[[1]] + 2, ix[[2]] + 1]) * (1 - fx[[2]]) +
        fx[[2]] * (field[[ix[[1]] + 1, ix[[2]] + 2] * (1 - fx[[1]]) +
        fx[[1]] * field[[ix[[1]] + 2, ix[[2]] + 2]));
      nv = part[[i]][[2]] + ax * dt; {part[[i]][[1]] + nv * dt, nv}, {i, Length[part]};

  ParticleManagePicker[in_] := Mod[(in[[1]][[1]] * nproc) / nx, nproc];

  ParticleManage[part_] := Module[{i, x, list}, list = Table[x = part[[i]][[1]];
    x = x + {If[x[[1]] < 0, +nx, If[x[[1]] ≥ nx, -nx, 0]}, If[x[[2]] < 0, +ny,
    If[x[[2]] ≥ ny, -ny, 0]}; {x, part[[i]][[2]]}, {i, Length[part]};

```

```

ElementManage[list, ParticleManagePicker];

DrawThree[speed_, rho_, potential_] := Module[{r, g, b, rt, gt, bt},
  mpiReduce[speed, rt, mpiSum, 0, mpiCommWorld];
  r = Transpose[MyNormalize[rt]];
  mpiReduce[rho, gt, mpiSum, 0, mpiCommWorld];
  g = Transpose[MyNormalize[gt]];
  mpiReduce[potential, bt, mpiSum, 0, mpiCommWorld];
  b = Transpose[MyNormalize[bt]]; If[idproc == 0, Show[Graphics[
    Raster[Table[List[PixelDouble[r, i, j], PixelDouble[g, i, j], PixelDouble[b, i, j]],
      {i, 2 * Length[r] - 1}, {j, 2 * Length[r][[1]] - 1}], AspectRatio -> Automatic]]];

MyNormalize[list_] := Module[{i, j, mx, mn}, mx = mn = list[[1, 1]];
  Do[If[list[[i, j]] > mx, mx = list[[i, j]]; If[list[[i, j]] < mn, mn = list[[i, j]]],
    {i, Length[list]}, {j, Length[list][[1]]}]; If[mx > mn,
    Table[(list[[i, j]] - mn) / (mx - mn), {i, Length[list]}, {j, Length[list]}, 0]];

PixelDouble[l_, i_, j_] := If[Mod[j, 2] < 1,
  If[Mod[i, 2] < 1, (1[[IntegerPart[(i - 1) / 2] + 1, IntegerPart[(j - 1) / 2] + 1]] +
    1[[1 + IntegerPart[(i - 1) / 2], 2 + IntegerPart[(j - 1) / 2]]) +
    1[[2 + IntegerPart[(i - 1) / 2], 1 + IntegerPart[(j - 1) / 2]]) +
    1[[2 + IntegerPart[(i - 1) / 2], 2 + IntegerPart[(j - 1) / 2]])] / 4,
  (1[[IntegerPart[(i - 1) / 2] + 1, IntegerPart[(j - 1) / 2] + 2]] +
    1[[IntegerPart[(i - 1) / 2] + 1, IntegerPart[(j - 1) / 2] + 1]]) / 2],
  If[Mod[i, 2] < 1, (1[[IntegerPart[(i - 1) / 2] + 1, IntegerPart[(j - 1) / 2] + 1]] +
    1[[2 + IntegerPart[(i - 1) / 2], 1 + IntegerPart[(j - 1) / 2]])] / 2,
  1[[IntegerPart[(i - 1) / 2] + 1, IntegerPart[(j - 1) / 2] + 1]]];

potkernel = Table[If[i == 1 && j == 1, I,
  If[j * 2 > ny, If[i * 2 > nx, 1 / ((nx + 1 - i) * (nx + 1 - i) + (ny + 1 - j) * (ny + 1 - j)),
    1 / ((i - 1) * (i - 1) + (ny + 1 - j) * (ny + 1 - j))],
  If[i * 2 > nx, 1 / ((nx + 1 - i) * (nx + 1 - i) + (j - 1) * (j - 1)),
    1 / ((i - 1) * (i - 1) + (j - 1) * (j - 1))]] + I], {i, nx}, {j, ny}];

fieldXkernel = Table[If[i == 1 && j == 1, 0, -I *
  If[j * 2 > ny, If[i * 2 > nx, (nx + 1 - i) / ((nx + 1 - i) * (nx + 1 - i) + (ny + 1 - j) * (ny + 1 - j)),
    (i - 1) / ((i - 1) * (i - 1) + (ny + 1 - j) * (ny + 1 - j))],
  If[i * 2 > nx, (nx + 1 - i) / ((nx + 1 - i) * (nx + 1 - i) + (j - 1) * (j - 1)),
    (i - 1) / ((i - 1) * (i - 1) + (j - 1) * (j - 1))]]], {i, nx}, {j, ny}];

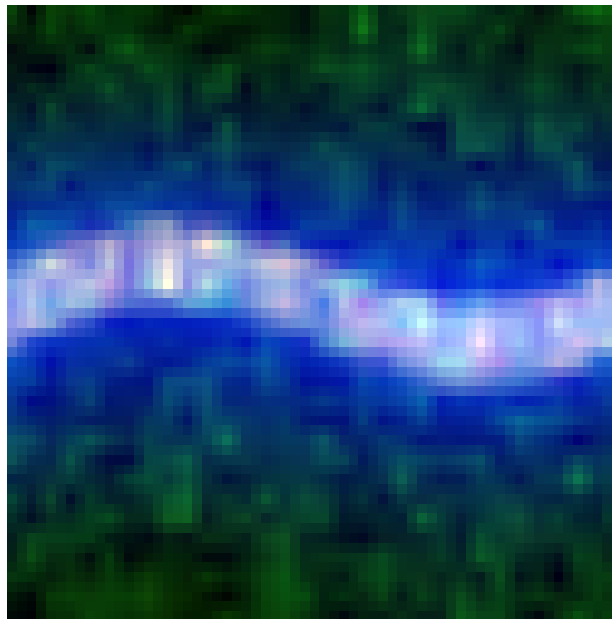
fieldYkernel = Table[If[i == 1 && j == 1, 0, -I *
  If[j * 2 > ny, If[i * 2 > nx, (ny + 1 - j) / ((nx + 1 - i) * (nx + 1 - i) + (ny + 1 - j) * (ny + 1 - j)),
    (ny + 1 - j) / ((i - 1) * (i - 1) + (ny + 1 - j) * (ny + 1 - j))],
  If[i * 2 > nx, (j - 1) / ((nx + 1 - i) * (nx + 1 - i) + (j - 1) * (j - 1)),
    (j - 1) / ((i - 1) * (i - 1) + (j - 1) * (j - 1))]]], {i, nx}, {j, ny}];

In[18]= particles =
  If[idproc == 0, Table[If[i > np, x = Random[]; {{x * nx, (ny / 2) + (Sin[x * Pi * 2] * ny / 16) +
    ((Random[] - 0.5) * ny / 8)}, {(Random[]) * 0.5, 0.5 * (Random[] - 0.5)}},
    {{Random[] * nx, Random[] * ny}, {(Random[] - 0.5) / 4, (Random[] - 0.5) / 4}},
    {i, np + nb}], List[]]; particles = ParticleManage[particles];

```

```
In[19]= {Timing[rho = ChargeDeposit[particles, charge]; field =
  FieldStitch[Re[FieldSolve[rho, fieldXkernel]], Re[FieldSolve[rho, fieldYkernel]]];
  potential = Re[FieldSolve[rho, potkernel]];
  particles = ParticleManage[ParticlePush[particles, Re[field]]];
  ke = KEDeposit[particles];], DrawThree[ke, rho, potential]}
```

```
Out[19]= {{0.614456, Null},
```



```
In[20]= {Timing[Do[rho = ChargeDeposit[particles, charge]; field =
  FieldStitch[Re[FieldSolve[rho, fieldXkernel]], Re[FieldSolve[rho, fieldYkernel]]];
  particles = ParticleManage[ParticlePush[particles, Re[field]], {4}];],
  ke = KEDeposit[particles]; potential = Re[FieldSolve[rho, potkernel]];
  DrawThree[ke, rho, potential]}
```

```
Out[20]= {{1.83851, Null},
```

