

# Virtual Cathode Simulation in 1D Using a Grid-Free Poisson Solver

**Ying Ki Yim**

Research Experience for Undergraduates Project

May 5 - June 30, 2006

Faculty Advisor: Lyudmyla Barannyk

University of Michigan  
Department of Mathematics  
Ann Arbor, MI

National Science Foundation

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Mathematical Formulation</b>	<b>3</b>
<b>3</b>	<b>Numerical Methods</b>	<b>6</b>
<b>4</b>	<b>Numerical Results</b>	<b>9</b>
4.1	Phase Plane for Various Injection Current Rates . . . . .	9
4.2	Number Density . . . . .	10
4.3	Convergence of Numerical Methods . . . . .	10
<b>5</b>	<b>Conclusions</b>	<b>13</b>
<b>6</b>	<b>Acknowledgments</b>	<b>13</b>
<b>A</b>	<b>APPENDIX: Codes</b>	<b>14</b>
	<b>References</b>	<b>20</b>

## List of Figures

1	Phase Plane at $t_f = 5$ with various $J$ ; direct summation(circle), GF(point)	10
2	$J = 5$ , Particle density $\frac{n}{n_0}$ versus $t$ at meshpoints $\frac{2}{17}, \frac{4}{17}, \dots, \frac{12}{17}$ ; direct summation(solid black), GF(dashed, gray) . . . . .	11
3	$J = 100$ , Particle density $\frac{n}{n_0}$ versus $t$ at meshpoints $\frac{2}{17}, \frac{4}{17}, \dots, \frac{12}{17}$ ; direct summation(solid black), GF(dashed, gray) . . . . .	11
4	Convergence of Euler's Method; loglog plot . . . . .	12
5	Convergence of Runge-Kutta 4th Order Method; loglog plot . . . . .	12

# 1 Introduction

Systems with virtual cathode are effective generators of electromagnetic pulses, and design of such systems often requires charged particle flow to be simulated with high accuracy. Recent works have incorporated the geometrical complexity of real devices into simulations [1], while others studied the effect of multiple velocity-beam to chaotic oscillations under a virtual cathode formation [2].

A problem that follows immediately is to develop an efficient method for such simulations. The particle-in-cell method is a traditional mesh-based method, widely used and effective in reducing the cost of force computation, however impractical to apply to complicated domains involving small scale features where exceedingly fine meshes are required [3]. As opposed to mesh-based, a common grid-free approach is to simulate such system from the Lagrangian perspectives as a cloud of charged particles, and obtaining the evolution by integrating particles forward in time by Newton's equation. The most expensive part of the process is the computation of particle forces. For such a system, the force  $F_i$  acting on particle  $x_i$  is directly related to the rest of the particles in the system. Since forces acting on each particle must be computed, the direct summation approach is an  $O(N^2)$  operation,  $N$  being the total number of particles in the system.

Various algorithms have been developed for improvement, of which Treecode algorithms, originally developed to compute gravitational forces in astrophysics, used also extensively in fluid dynamics and molecular dynamics, was adapted as a grid-free alternative [4], in which particle-particle interactions are replaced by suitably chosen particle-cluster interactions, hence reducing the number of operations.

In this paper, we restrict our attention to applying an alternative grid-free way of computing forces using the simple form of Green's function via a sorting algorithm.

## 2 Mathematical Formulation

In 1-Dimension, two infinite plates are located at  $x = a, b$  with voltage  $V_a$  and  $V_b$  respectively. Initially the gap between two plates is a vacuum, then one of the plates is heated until electrons are emitted from the plate, thus a current flows through the gap. As the emission process reaches a sufficient rate, the current between the plates approaches a limiting value  $J_{CL}$ , predicted by Child-Langmuir-Law, emitted charged particles begin to turn around and hit the emitting plane. This will give rise to the potential minimum and will allow the charged particles to cross the gap again. Hence the process repeats. In this paper, we will examine the classical formulation in which the two infinite plates are held at the same potential, and emitted particles are given some initial velocity  $v$  [5].

The total potential  $\Phi(x)$  satisfies the Poisson equation

$$\frac{d^2\Phi}{dx^2} = -\frac{\rho}{\epsilon_0}$$

subject to boundary conditions

$$\Phi(a) = V_a, \quad \Phi(b) = V_b$$

where  $\rho(x)$  is the charge density;  $x = a$  and  $x = b$  are the left and right endpoints of the gap between two plates,  $\varepsilon_0$  is the permittivity of free space.

Let the number of electrons between endpoints be  $N$ . The electrons with charges  $q_j$  are located at points  $x_j$ ,  $j = 1, \dots, N$ . The charge density is

$$\rho(x) = \sum_{j=1}^N \delta(x - x_j) q_j$$

where  $\delta(x)$  is the Dirac delta function. The total potential could be written as

$$\Phi(x) = \Phi_H(x) + \Phi_P(x)$$

where  $\Phi_H(x)$  is the the general solution of the homogeneous equation

$$\frac{d^2 \Phi_H}{dx^2} = 0$$

and  $\Phi_P(x)$  is a particular solution of

$$\frac{d^2 \Phi_P}{dx^2} = -\frac{\rho}{\varepsilon_0}$$

The particular solution  $\Phi_P(x)$  is chosen to be

$$\Phi_P(x) = \frac{1}{\varepsilon_0} \sum_{j=1}^N G(x - x_j) q_j$$

where  $G(x, y) = -\frac{1}{2}|x - y|$  is the free space Green's function for 1-D Poisson equation. The homogeneous part of the solution takes the form of

$$\Phi_H(x) = Ax + B$$

Constants  $A$  and  $B$  are obtained by imposing the boundary conditions on the solution  $\Phi(x)$

$$\begin{cases} \Phi(a) = \Phi_H(a) + \Phi_P(a) = V_a = A(a) + B + \Phi_P(a) \\ \Phi(b) = \Phi_H(b) + \Phi_P(b) = V_b = A(b) + B + \Phi_P(b) \end{cases}$$

$$V_a - V_b = A(a - b) + \Phi_P(a) - \Phi_P(b)$$

$$A = \frac{1}{a - b} (V_a - V_b - \Phi_P(a) + \Phi_P(b)) = \frac{1}{b - a} (V_b - V_a + \Phi_P(a) - \Phi_P(b))$$

and

$$V_a - V_b \frac{a}{b} = B \left(1 - \frac{a}{b}\right) + \Phi_P(a) - \Phi_P(b) \frac{a}{b}$$

$$B \left(1 - \frac{a}{b}\right) = V_a - V_b \frac{a}{b} - \Phi_P(a) + \Phi_P(b) \frac{a}{b}$$

$$B = \frac{b}{b-a} (V_a - \Phi_P(a)) - \frac{a}{b-a} (V_b - \Phi_P(b))$$

By definition of the electric field,

$$\vec{F} = -q\vec{E}, \quad \vec{E} = \frac{\vec{V}}{m} = \frac{d\Phi}{dx}$$

The induced force on particle  $i$  is then given by

$$F_i = -q_i \left( \frac{d\Phi_H(x_i)}{dx} + \frac{d\Phi_P(x_i)}{dx} \right)$$

The particular component of the force acting on a particle  $x_i$  is

$$F_{iP}(x) = -q_i \frac{d\Phi_P(x_i)}{dx} = -\frac{q_i}{\varepsilon_0} \sum_{\substack{j=1 \\ j \neq i}}^N \frac{\partial}{\partial x} G(x_i, x_j) q_j$$

Note that

$$G(x, x_j) = -\frac{1}{2}|x - x_j| = \begin{cases} -\frac{1}{2}(x - x_j), & x > x_j \\ \frac{1}{2}(x - x_j), & x < x_j \end{cases}$$

we have

$$\frac{\partial}{\partial x} G(x, x_j) = \begin{cases} -\frac{1}{2}, & x > x_j \\ \frac{1}{2}, & x < x_j \end{cases} = H(x - x_j) - \frac{1}{2}$$

where  $H(x - x_j)$  is the Heaviside function or the unit step function. The force due to the particular solution can either be computed by direct summation using the above equations, which is  $O(N^2)$  operations, or we can rewrite its expression by using a simple form of the Green's function, which would reduce the number of operations significantly.

Let  $n_R$  and  $n_L$  be the number of particles to the right and to the left from  $x_i$ , respectively, when all particles are ordered in the manner such that

$$x(i) \leq x(j), \quad i \leq j$$

Then  $n_R = N - i$  and  $n_L = i - 1$ . The expression for the particular component of force can be written as

$$\begin{aligned} F_{iP} &= -\frac{q_i}{\varepsilon_0} \sum_{\substack{j=1 \\ j \neq i}}^N \frac{\partial}{\partial x} G(x_i, x_j) q_j = -\frac{q_i}{\varepsilon_0} \left[ \sum_{\substack{j \\ x_i > x_j \\ n_L}} \frac{\partial}{\partial x} G(x_i, x_j) q_j + \sum_{\substack{j \\ x_i < x_j \\ n_R}} \frac{\partial}{\partial x} G(x_i, x_j) q_j \right] \\ &= -\frac{q_i}{\varepsilon_0} \left[ -\frac{1}{2} \sum_{\substack{j \\ x_i > x_j \\ n_L}} q_j + \frac{1}{2} \sum_{\substack{j \\ x_i < x_j \\ n_R}} q_j \right] \end{aligned}$$

If we assume all particles have the same charge  $q$ , then  $F_{iP}$  can be simplified to

$$F_{iP} = \frac{q^2}{2\epsilon_0} (2i - N - 1)$$

The force due to the homogeneous solution at  $x_i$  is

$$F_{iH} = -q_i \frac{d\Phi_H(x_i)}{dx} = -q_i \frac{d}{dx} (A(x_i) + B) = -q_i A = -qA$$

when charges are the same. The constant

$$A = \frac{1}{b-a} (V_b - V_a + \Phi_P(a) - \Phi_P(b))$$

has components of the particular solution evaluated at endpoints. This can be simplified using the specific form of the Green's function

$$\Phi_P(x) = \frac{1}{\epsilon_0} \sum_{j=1}^N G(x - x_j) q_j = \frac{1}{\epsilon_0} \sum_{j=1}^N \left( -\frac{1}{2} \right) |x - x_j| q_j$$

If  $x = a$ , then  $a \leq x_j$  for all  $j = 1, \dots, N$ , hence,  $|a - x_j| = x_j - a$ . Similarly, if  $x = b$ , then  $b \geq x_j$  for all  $j$  and  $|b - x_j| = b - x_j$ . Therefore,

$$\Phi_P(a) = \frac{-1}{2\epsilon_0} \sum_{j=1}^N (x_j - a) q_j = \frac{-q}{2\epsilon_0} \sum_{j=1}^N (x_j - a) = \frac{-q}{2\epsilon_0} \left[ \sum_{j=1}^N x_j - aN \right]$$

$$\Phi_P(b) = \frac{-1}{2\epsilon_0} \sum_{j=1}^N (b - x_j) q_j = \frac{-q}{2\epsilon_0} \sum_{j=1}^N (b - x_j) = \frac{-q}{2\epsilon_0} \left[ bN - \sum_{j=1}^N x_j \right].$$

If all particles bare the same charge, the induced force on particle  $i$  is

$$\begin{aligned} F_i &= F_{iH} + F_{iP} \\ &= -\frac{q}{b-a} \left( V_b - V_a + \frac{-q}{2\epsilon_0} \left[ \sum_{j=1}^N x_j - aN \right] - \frac{-q}{2\epsilon_0} \left[ bN - \sum_{j=1}^N x_j \right] \right) + \frac{q^2}{2\epsilon_0} (2i - N - 1) \end{aligned}$$

□

### 3 Numerical Methods

With the force known, the particles are adverted by Newton's second law of motion,  $m_i \ddot{x}_i = F_i$ , where  $m_i$  is mass of particle  $i$ . Numerically, this is done by applying a time integration method to the first order system

$$\begin{cases} m_i \dot{v}_i = F_i \\ \dot{x}_i = v_i \end{cases} \quad (1)$$

This system can be solved by various numerical methods, of which we used Euler's method and Runge-Kutta 4th order method.

Consider the first order initial value problems of the following form

$$\frac{dy}{dt} = f(t, y(t)), \quad y(t_0) = y_0$$

Euler's method is a simple one-step method to approximate the solution. Let  $\Delta t = (t_f - t_0)/N_t$  be the step size,  $t_f$ ,  $t_0$  and  $N_t$  being the final time, initial time and the number of intervals respectively. The approximated solution  $w$  is sought at equally spaced intervals with

$$w_{i+1} = w_i + \Delta t f(t_i, w_i), \quad w_0 = y_0, \quad t_i = t_0 + i\Delta t \quad (i = 0, 1, 2, \dots, N_t)$$

The method is derived by expanding the exact solution in a Taylor series about the point  $t = t_i$ , producing

$$y(t) = y_i + (t - t_i)y'_i + \frac{1}{2}(t - t_i)^2 y''(\xi)$$

where  $\xi$  is guaranteed to lie between  $t$  and  $t_i$ . Evaluating the above expansion at  $t = t_i$  and substituting for  $y'_i$  with the right hand side of the differential equation, we obtain

$$y_{i+1} = y_i + \Delta t f(t_i, y_i) + \frac{1}{2}\Delta t^2 y''(\xi)$$

Euler's method arises by dropping the local truncation error term  $\frac{1}{2}\Delta t^2 y''(\xi)$  and replacing  $y_i$  by  $w_i$ .

Euler's method is by far the most simple method to implement, however it is also not very efficient and in particular its global error to the exact solution are  $O(\Delta t)$ . Note that the local truncation error measures how well the continuous equation is has been approximated by the discrete difference equation, and the global discretization measures how well the true solution has been approximated.

Higher order one-step methods can be constructed using the derivatives of  $f$  of the appropriate order. The Taylor methods are derived using the same approach as the derivation of the Euler's method, but assuming the exact solution has more continuous derivatives and thus retaining more terms of the Taylor expansion accordingly. In particular, the fourth order Taylor method is

$$w_{i+1} = w_i + h f(t_i, w_i) + \frac{\Delta t^2}{2} \frac{df}{dt}(t_i, w_i) + \frac{\Delta t^3}{6} \frac{d^2 f}{dt^2}(t_i, w_i) + \frac{\Delta t^4}{24} \frac{d^3 f}{dt^3}(t_i, w_i)$$

Although more accurate, the disadvantage associated with Taylor methods of order  $n > 1$  is that the derivatives of  $f$  are required. High order one-step method can be derived by approximating the right hand side of the Taylor methods. These methods are known as the Runge-Kutta methods.

Runge-Kutta methods use exclusively the values of  $f$  to obtain the approximation of the appropriate order. In the fourth order case, the most common scheme updates the approximate solution at each time step according to the formula

$$w_{i+1} = w_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4),$$

where

$$\begin{aligned} k_1 &= \Delta t f(t_i, w_i) \\ k_2 &= \Delta t f\left(t_i + \frac{\Delta t}{2}, w_i + \frac{k_1}{2}\right) \\ k_3 &= \Delta t f\left(t_i + \frac{\Delta t}{2}, w_i + \frac{k_2}{2}\right) \\ k_4 &= \Delta t f(t_i + \Delta t, w_i + k_3) \end{aligned}$$

The method requires the four function evaluation per time step and its global discretization error is of  $O(\Delta t^4)$  [6].

The system (1) can be written as

$$\mathbf{u}' = f(\mathbf{u}, \mathbf{F}), \quad \mathbf{u} = \begin{pmatrix} \mathbf{x} \\ \mathbf{v} \end{pmatrix}$$

which is a first order system with  $\mathbf{u}$  being a 2 by  $N$  matrix,  $\mathbf{x}$ ,  $\mathbf{v}$  and  $\mathbf{F}$  being the position, velocity and force arrays of particles respectively,  $N$  being the length of arrays. We have

$$\frac{dx}{dt} = v, \quad \frac{dv}{dt} = \frac{F}{m}$$

thus

$$\mathbf{u}' = \begin{pmatrix} \mathbf{x}' \\ \mathbf{v}' \end{pmatrix} = f\left(\begin{pmatrix} \mathbf{x} \\ \mathbf{v} \end{pmatrix}, \mathbf{F}\right) = \begin{pmatrix} \mathbf{v} \\ \frac{\mathbf{F}}{m_i} \end{pmatrix}$$

The numerical approximation of the system will then be, with  $\mathbf{w}_i$  as the approximate to the exact solution  $\mathbf{u}_i$

$$\mathbf{w}_{i+1} = \mathbf{w}_i + \Delta t f(\mathbf{w}_i, \mathbf{F})$$

being the Euler's method and

$$\mathbf{w}_{i+1} = \mathbf{w}_i + \frac{1}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4),$$

where

$$\begin{aligned} \mathbf{k}_1 &= f(\mathbf{u}, \mathbf{F}_1), & \mathbf{F}_1 &= \mathbf{F}(\mathbf{x}) \\ \mathbf{k}_2 &= f\left(\mathbf{u} + \Delta t \frac{\mathbf{k}_1}{2}, \mathbf{F}_2\right), & \mathbf{F}_2 &= \mathbf{F}\left(\mathbf{x} + \Delta t \frac{\mathbf{k}_{1x}}{2}\right) \\ \mathbf{k}_3 &= f\left(\mathbf{u} + \Delta t \frac{\mathbf{k}_2}{2}, \mathbf{F}_3\right), & \mathbf{F}_3 &= \mathbf{F}\left(\mathbf{x} + \Delta t \frac{\mathbf{k}_{2x}}{2}\right) \\ \mathbf{k}_4 &= f(\mathbf{u} + \Delta t \mathbf{k}_3, \mathbf{F}_4), & \mathbf{F}_4 &= \mathbf{F}(\mathbf{x} + \Delta t \mathbf{k}_{3x}) \end{aligned}$$

the fourth order Runge-Kutta method. Note that due to the change in  $\mathbf{x}$ , force  $\mathbf{F}$  has to be reevaluated not only at each time step, but also at each stage of the Runge-Kutta method in order to compute  $\mathbf{k}_{1,2,3,4}$ .

□

## 4 Numerical Results

### 4.1 Phase Plane for Various Injection Current Rates

We begin by studying the phase plane. Denote the injection current (or emission rate) at  $x = a$  by  $J$ . Then  $J = env$ , where  $e$  is the charge of an electron,  $n$  is the electron number density and  $v$  is the velocity of the electron. Consider

$$q = en\Delta x$$

the charge of a macro-particle or the total charge of a stream of electrons of length  $\Delta x$ . Then the charge of an electron can be written as

$$e = \frac{q}{n\Delta x}$$

Since  $\Delta x = \Delta t \cdot v$ , velocity  $v$  can be expressed as

$$v = \frac{\Delta x}{\Delta t}$$

The injection current  $J$  can then be written as

$$J = env = \frac{q}{n\Delta x} n \frac{\Delta x}{\Delta t} = \frac{q}{\Delta t} \tag{2}$$

Given an injection current and time step  $\Delta t$ , we compute the charge of a macro-particle  $q = J\Delta t$  by relation (2).

In numerical simulation of an one dimensional virtual cathode we set mass of each particle  $m = 1$  and permittivity  $\varepsilon_0 = 1$ . Using Euler's method as the time integrating method and  $\Delta t = 0.01$ , we examine the phase plane of the virtual cathode at  $t_f = 5$  for various injection current  $J = 5, 10, 12, 14, 15, 20, 50, 100$  using direct summation and the simple form of Green's function (GF). For small values of injection current  $J$ , particles travel across the gap, forming a parabolic concave up profile where concavity increases with the value of  $J$ . For sufficiently small  $J$ 's, i.e.  $J < 10$ , the curvature is very much close to zero, and as  $J$  increases, the curvature increases until  $J$  reaches a limiting value,  $J_{CL} = 15$ , particles begin to turn around and strike the emitting plane. As  $J$  increases further, the location  $x$  where the particles turn around approaches the boundary  $x = a$  (see Figure 1).

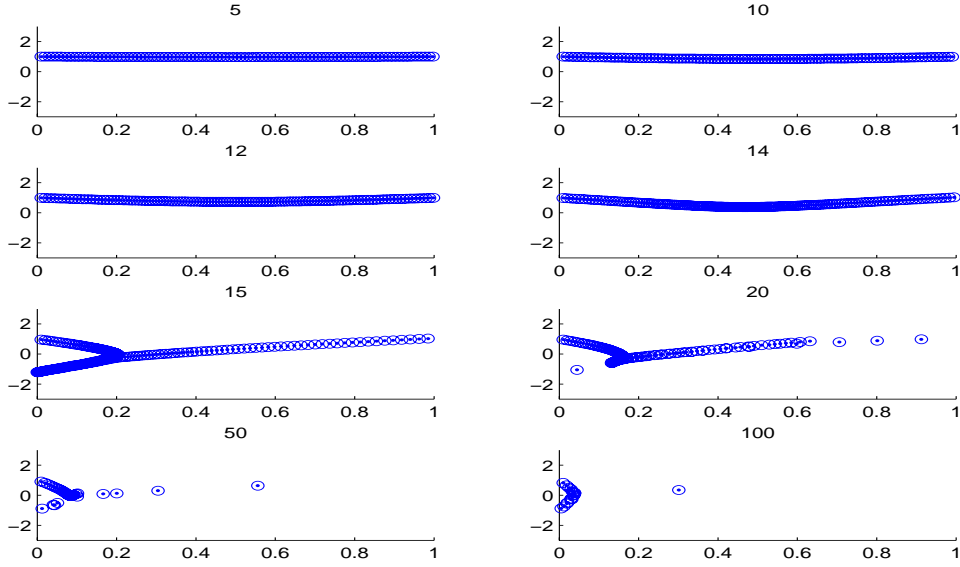


Figure 1: Phase Plane at  $t_f = 5$  with various  $J$ ; direct summation(circle), GF(point)

## 4.2 Number Density

The evolution of the normalized number electron density  $n/n_0$  using direct summation and the simple form of Green's function is also investigated, where  $n_0$  is the maximum number of particles in the system. The number density is obtained by the standard *area weighting method* used in the Particle-In-Cell Method (PIC).

Let a particle  $j$  be located at  $x_j$ . We measure the number density at points  $a = L_0 < L_1, \dots < L_{nd-1} < L_{nd} = b$ . Assume that the particle is located between  $L_i$  and  $L_{i+1}$ , i.e.  $L_i \leq x_j \leq L_{i+1}$ . Denote by  $\Delta L$  the distance between  $L_i$  and  $L_{i+1}$ , and  $n_i$  the number density at point  $L_i$ . Then the contribution of point  $j$  to  $n_i$  and  $n_{i+1}$  is

$$q_j \frac{L_{i+1} - x_j}{\Delta L}, \quad q_j \frac{x_j - L_i}{\Delta L}$$

respectively.

Using 4th order Runge-Kutta method and  $\Delta t = 0.05$ , we examine the particle density at 16 different mesh points from  $t_0 = 0$  to  $t_f = 5$ . For small values of  $J$ , the number density at the mesh points attains a steady state. For increasing values of  $J$  greater than the critical value  $J_{CL}$ , the number densities begin to oscillate at each meshpoint. Oscillation tends to decrease as the location of the meshpoint is further away from the boundary  $x = a$  (see Figure 2 and 3).

## 4.3 Convergence of Numerical Methods

We verified the convergence of the two methods. The global discretization error  $\|e\|$  of a method is defined as  $\|\text{exact solution} - \text{approximate solution}\|$ , and for a particular method of order  $p$ ,

$$\|e\| = C \Delta t^p,$$

$$\ln \|e\| = \ln C + p \ln \Delta t$$

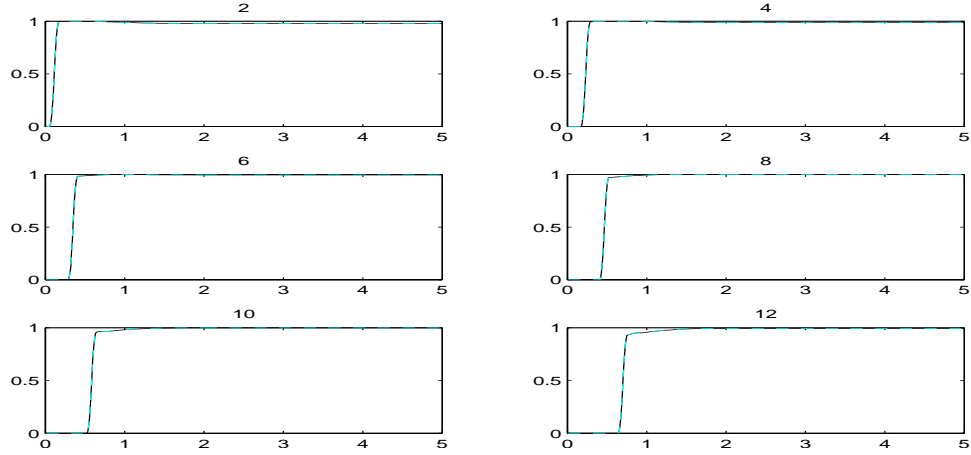


Figure 2:  $J = 5$ , Particle density  $\frac{n}{n_0}$  versus  $t$  at meshpoints  $\frac{2}{17}, \frac{4}{17}, \dots, \frac{12}{17}$ ; direct summation(solid black), GF(dashed, gray)

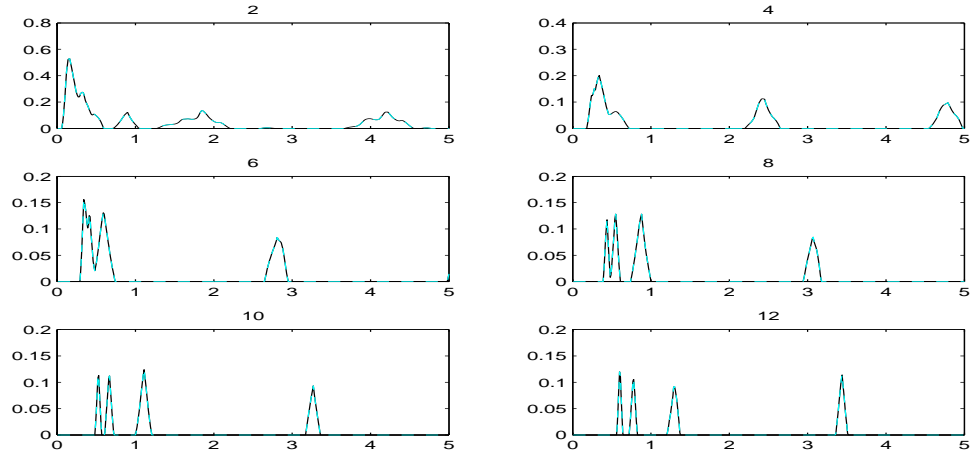


Figure 3:  $J = 100$ , Particle density  $\frac{n}{n_0}$  versus  $t$  at meshpoints  $\frac{2}{17}, \frac{4}{17}, \dots, \frac{12}{17}$ ; direct summation(solid black), GF(dashed, gray)

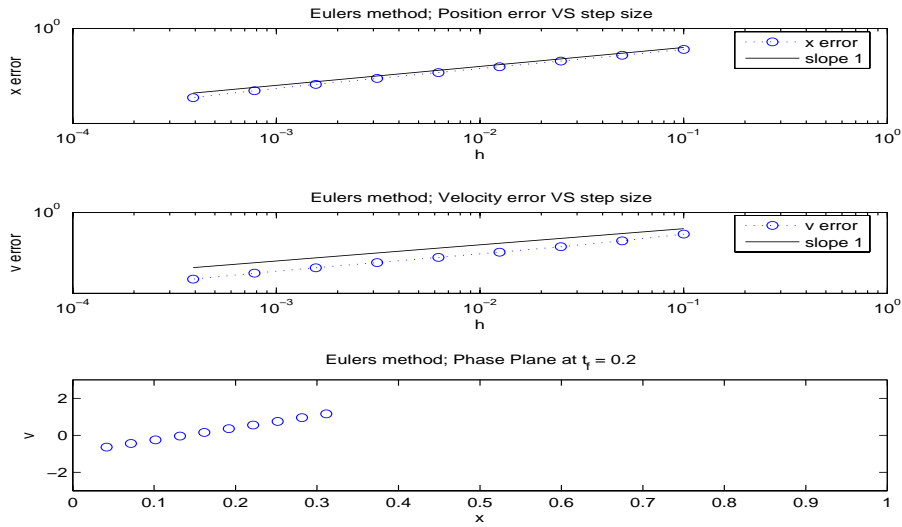


Figure 4: Convergence of Euler's Method; loglog plot

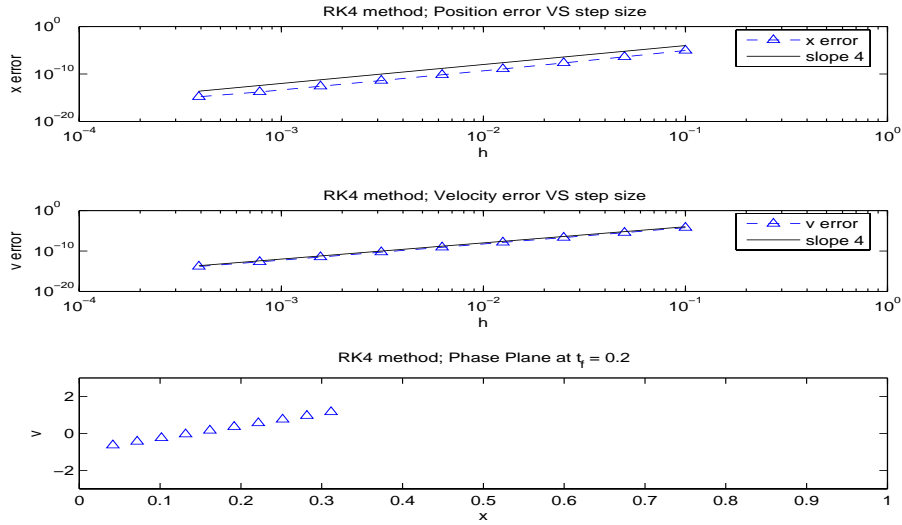


Figure 5: Convergence of Runge-Kutta 4th Order Method; loglog plot

in a loglog plot  $\|e\|$  VS  $\Delta t$  should approximately have a slope of  $p$ , here  $\|\cdot\|$  is a norm.

We first obtained a reference solution to the exact solution by using a time-step of  $\frac{0.1}{2^{10}}$  since the actual solution is not available. Using 10 particles initially in the system, and with a final step that no particles hit either planes, we examine the infinite norm of the error between the reference solution and the solution at various time steps of  $\frac{0.1}{2^k}$ , where  $k = 0, 1, \dots, 8$  using a loglog plot, and indeed we achieve linear convergence and quartic convergence for Euler's Method and RK4 method, respectively (see Figure 4 and 5).

## 5 Conclusions

We studied numerically the formation of an one dimensional Virtual Cathode using Euler's method and Runge-Kutta 4th order method.

An algorithm that uses the simple form of Green's function in computing the electrostatic forces in charged particle simulations is presented. Since particles are emitted with a fixed velocity and at a fixed position, it is safe to update the position of the particles by shifting the indices by one for low injection current  $J$  under the critical value  $J_{CL}$  where particles travel across the gap in a parabolic profile. In such cases, one can compute the force acting on a particle directly with its position in the system. The number of operations will then be  $O(N)$  as opposed to  $O(N^2)$  with direct summation. For injection current higher than  $J_{CL}$ , which is the case of interest, a sorting algorithm is used to compute the positions of particles in the system. The number of operations will then depend on the sorting algorithm. We use Quick Sort algorithm that is  $O(N \log N)$ . For the comparison, the Bubble Sort algorithm requires  $O(N^2)$  operations.

Phase planes for various injection current  $J$  and the evolution of number density at various mesh points are investigated and simulated by both direct summation and the simple form of Green's function via sorting algorithm. Results of the two algorithms are compared and they are consistent as expected. Convergence of the two numerical methods is also verified.

## 6 Acknowledgments

The author expresses his deepest gratitude to Professor L. Barannyk who spent time and patience guiding the author through the topic. He would also like to further extend his appreciation to the University of Michigan and the National Science Foundation for organizing the Research Experience for Undergraduates Program.

## A APPENDIX: Codes

```
%-----  
function conv [e_x_euler,e_v_euler,e_F_euler] =  
trial_injection(0.1/2^10,1); [e_x_rk4, e_v_rk4, e_F_rk4] =  
trial_injection(0.1/2^10,2); e_x_error = []; e_v_error = [];  
e_F_error = []; r_x_error = []; r_v_error = []; r_F_error = []; t =  
[]; for i= 1:2  
    h = 0.1;  
    while h > 0.1/2^9  
        [a,b,c] = trial_injection(h,i);  
        if i==1  
            e_x_error = [e_x_error; norm(e_x_euler-a, inf)];  
            e_v_error = [e_v_error; norm(e_v_euler-b, inf)];  
            e_F_error = [e_F_error; norm(e_F_euler-c, inf)];  
            t = [t h]; h = h/2;  
        end  
        if i==2  
            r_x_error = [r_x_error; norm(e_x_rk4-a, inf)];  
            r_v_error = [r_v_error; norm(e_v_rk4-b, inf)];  
            r_F_error = [r_F_error; norm(e_F_rk4-c, inf)];  
            h = h/2;  
        end  
    end  
end  
end loglog(t,e_x_error,'o:');axis([1e-4,1,1e-5,1]); hold on;  
loglog(t,t,'k'); loglog(t,e_v_error,'o:'); axis([1e-4,1,1e-5,1]);  
hold on; loglog(t,t,'k'); plot(e_x_euler, e_v_euler, 'o');  
axis([0,1,-3,3]); hold off; figure; loglog(t,r_x_error,'^--');  
axis([1e-4,1,1e-20,1]); hold on; loglog(t,t.^4,'k');  
loglog(t,r_v_error,'^--'); axis([1e-4,1,1e-20,1]); hold on;  
loglog(t,t.^4,'k'); plot(e_x_rk4, e_v_rk4, '^'); axis([0,1,-3,3]);  
hold off;  
%-----  
function [x,v,F] = trial_injection(h,m)  
%h = step size, m = method  
global e_0 m_e q_e a b Va Vb e_0 = 1;m_e = 1;q_e = 1; a = 0; b = 1;  
Va = 1; Vb = 1;  
  
x = []; for n = 1:10  
    x = [x 0.01*n];  
end v = ones(1,length(x));F = F_i(x);  
  
tf = 0.2; if m == 1  
    for t = 1:(floor(tf/h))  
        [x,v] = euler(x,v,F,h);  
        F = F_i(x);  
    end  
end
```

```

else
    for t = 1:(floor(tf/h))
        [x,v] = rkf(x,v,F,h);
        F = F_i(x);
    end
end
%-----
function var_J

j = [5,10,12,14,15,20,50,100];
for n = 1:length(j)
    subplot(4,2,n); title(j(n));
    emission(j(n),0.01,1)
end
%-----
function emission(J,h,m)
%J = injection current, h = step size, m = method
global e_0 m_e q_e a b Va Vb e_0 = 1; m_e = 1; q_e = J*h;
a = 0; b = 1;
Va = 1; Vb = 1;

x = []; v = []; F = F_i(x);
plot(x,v,'ro'); axis([a,b,-3,3]); hold on;

tf = 5; n_den = [];
if m == 1
    for t = 1:(floor(tf/h))
        x = [x 0]; v = [v 1];
        F = F_i(x);
        [x,v] = euler(x,v,F,h);
        [x,v] = absorb(x,v);
        F = F_i(x);
    end
else
    for t = 1:(floor(tf/h))
        x = [x 0]; v = [v 1];
        F = F_i(x);
        [x,v] = rkf(x,v,F,h);
        [x,v] = absorb(x,v);
        F = F_i(x);
    end
end plot(x,v,'o');
%-----
function n_den_var_J

j = [5,100];
n_density(j(1),0.01,2,16); figure;

```

```

n_density(j(2),0.01,2,16);
%-----
function n_density(J,h,m,l)
%l = number of mesh points
global e_0 m_e q_e a b Va Vb
e_0 = 1; m_e = 1; q_e = J*h;
a = 0; b = 1;
Va = 1; Vb = 1;

x = []; v = []; F = F_i(x);

tf = 5; n_den = []; time = [];
if m == 1
    for t = 1:(floor(tf/h))
        x = [x 0]; v = [v 1];
        F = F_i(x);
        [x,v] = euler(x,v,F,h);
        [x,v] = absorb(x,v);
        F = F_i(x);
        if length(x)>0
            n_temp = awm(x,1/(l+1));
            n_den = [n_den; n_temp];
        end
        time = [time t];
    end
else
    for t = 1:(floor(tf/h))
        x = [x 0]; v = [v 1];
        F = F_i(x);
        [x,v] = rkf(x,v,F,h);
        [x,v] = absorb(x,v);
        F = F_i(x);
        if length(x)>0
            n_temp = awm(x,1/(l+1));
            n_den = [n_den; n_temp];
        end
        time = [time t];
    end
end
for p =1:l
    if (p==2 | p==4 | p==6 | p==8 | p==10 | p==12)
        subplot(3,2,(p/2));plot((time*h),n_den(:,p));
        title(p);hold on;
    end
end
%-----
function F = F_s(x)

```

```

%compute force using summation
global e_0 m_e q_e a b Va Vb

A = q_e/(b-a);
F_h = A*(Vb-Va+(-q_e)/(2*e_0)*(sum(x)-a*length(x))-(-q_e)/(2*e_0)*(b*length(x)-sum(x)));
F_p = zeros(1,length(x)); i = 1; for i = 1:length(x)
    for j = 1:length(x)
        if x(j)>x(i)
            F_p(i) = F_p(i) - q_e^2/(2*e_0);
        end
        if x(j)<x(i)
            F_p(i) = F_p(i) + q_e^2/(2*e_0);
        end
    end
end
end
F = F_p - F_h;
%-----
function F = F_i(x)
%compute force using indices array
global e_0 m_e q_e a b Va Vb

A = q_e/(b-a);
F_h = A*(Vb-Va+(-q_e)/(2*e_0)*(sum(x)-a*length(x))-(-q_e)/(2*e_0)*(b*length(x)-sum(x)));
F_p = zeros(1,length(x)); u = [1:length(x)]; u_s = quick_sort(x);
for n = 1:length(x)
    i = 1;
    while u_s(i)~=u(n)
        i = i + 1;
    end
    F_p(n) = q_e^2/(2*e_0)*(2*i-length(x)-1);
end
F = F_p - F_h;
for n = 1:length(x)
    s = [];
    for m = 1:length(x)
        if x(n) == x(m)
            s = [s m];
        end
    end
end
if length(s) > 1
    F_same = 0;
    for k = 1:length(s)
        F_same = F_same + F(s(k));
    end
    F_same = F_same/length(s);
    for k = 1:length(s)
        F(s(k)) = F_same;
    end
end

```

```

        end
    end
end
%-----
function [x,v] = euler(x,v,F,h)
%using euler's method to propagate electrons
global e_0 m_e q_e

[xp,vp] = f(x,v,F);
x = x + h*xp;
v = v + h*vp;
%-----
function [x,v] = rkf(x,v,F,h)
%using RK4 method to propagate electrons
global e_0 m_e q_e

[k1x,k1v] = f(x,v,F); F = F_i(x+h*k1x/2);
[k2x,k2v] = f(x+h*k1x/2,v+h*k1v/2,F); F = F_i(x+h*k2x/2);
[k3x,k3v] = f(x+h*k2x/2,v+h*k2v/2,F); F = F_i(x+h*k3x);
[k4x,k4v] = f(x+h*k3x,v+h*k3v,F);

x = x + h*(k1x+2*k2x+2*k3x+k4x)/6;
v = v + h*(k1v+2*k2v+2*k3v+k4v)/6;
%-----
function [dx,dv] = f(x,v,F)
%modify x and v
global e_0 m_e q_e

dx(1:length(x)) = v;
dv(1:length(v)) = F/m_e;
%-----
function [xp,vp] = absorb(x,v)
%eliminate absorbed electrons from vectors
global e_0 m_e q_e a b
xp = []; vp = [];
for i = 1:length(x)
    if (x(i)>=a)&(x(i)<=b)
        xp = [xp x(i)];
        vp = [vp v(i)];
    end
end
end
%-----
function n_c = awm(x,l)
%area weighting method
global e_0 m_e q_e a b
d = floor((b-a)/l); n = [];
for k = 1:d

```

```

        n = [n k*1];
    end
    if rem((b-a),l)==0
        n = n(1:(length(n)-1));
        n_c = zeros(1,floor((b-a)/l)-1);
    else
        n_c = zeros(1,floor((b-a)/l));
    end
    for i = 1:length(x)
        j = floor(x(i)/l);
        if j~=0 & j~=length(n)
            n_c(j) = n_c(j) + q_e*(n(j+1)-x(i))/l;
            n_c(j+1) = n_c(j+1) + q_e*(x(i)-n(j))/l;
        elseif j==0
            n_c(j+1) = n_c(j+1) + q_e*(x(i)-a)/l;
        else
            n_c(j) = n_c(j) + q_e*(b-x(i))/l;
        end
    end
    n_c = n_c./norm(n_c, inf);
%-----
function [u] = quick_sort(x)
%quick sort input array x into ascending order, return the sorted indices array u

if length(x)>1
    v = x;
    u = [1:length(x)];
    [v,u] = q_sort(v,1,length(v),u);
else
    u = [1];
end
%-----
function [v,u] = q_sort(v, left, right, u)

l_hold = left; r_hold = right; pivot = v(left); u_pivot = u(left);
while left<right
    while (v(right)>=pivot) && (left<right)
        right = right - 1;
    end
    if left~=right
        v(left) = v(right);
        u(left) = u(right);
        left = left + 1;
    end
    while (v(left)<=pivot) && (left<right)
        left = left + 1;
    end
    if left~=right

```

```

        v(right) = v(left);
        u(right) = u(left);
        right = right - 1;
    end
end
v(left) = pivot; u(left) = u_pivot;
pivot = left;
left = l_hold; right = r_hold;
if left<pivot
    [v,u] = q_sort(v,left,(pivot-1),u);
end if right>pivot
    [v,u] = q_sort(v,(pivot+1),right,u);
end
%-----

```

## References

- [1] Assous F. *Numerical Simulation of a 3D virtual cathode oscillator* WIT TRANSACTIONS ON MODELLING AND SIMULATION, VOL 42, 2006
- [2] Kalinin Yu.A., Hramov A.E. *Experimental and Theoretical Investigation into the Effect of Electron Velocity Distribution on Chaotic Oscillations in an Electron Beam under Virtual Cathode Formation Conditions* TECHNICAL PHYSICS , VOL 51, NO. 5, 558-566, 2006
- [3] Birdsall C.K., Langdon A.B. *Plasma Physics via Computer Simulation* Bristol, U.K.: IOP, 1991
- [4] Christlieb A.J., Krasny R., Verboncoeur J.P. *Efficient particle simulation of a virtual cathode using a grid-free treecode Poisson solver* IEEE TRANSACTIONS ON PLASMA SCIENCE **32** (2): 384-389 Part 1, 2004
- [5] Birdsall C.K., Bridges W.B. *Electron Dynamics of Diode Regions* ELECTRICAL SCIENCE SERIES, 69-71, New York: Academic, 1966.
- [6] Bradie B. *A Friendly Introduction to Numerical Analysis* PEARSON PRENTICE HALL, **Chp 7**: 546-575, 2006