The Design and Implementation of Atomic Ion Shuttling Protocols in a Multi-dimensional Ion Trap Array

Mark Yeo
University of Michigan, Department of Physics, Address
Ann Arbor, MI 48109, USA

A senior honors thesis submitted to the Department of Physics of the University of Michigan Ann Arbor in submission for the 2006 W.L. Williams Award and in partial fulfillment of the Degree of Honors Bachelor's of Science in Physics.

May 4, 2006

Abstract

Harnessing the laws of Quantum Mechanics, Quantum Computers can potentially outstrip the performance of classical computers.

The ion trap is a promising scheme for quantum computation. Ions are trapped via a combination of static and oscillating electric fields and each quantum bit is stored in the hyperfine level of an atomic ion. However, due to the extremely delicate nature of quantum computing, severe technical problems have limited the demonstrations of quantum computation protocols to only a few qubits(<10).

This thesis documents major steps made towards scaling the ion trap quantum computer to a large number of qubits. Laser cooled atomic ions are stored shuttled and swapped in a 2-dimensional, 11-zone ion trap array. The trap consists of two linear rf-trap sections joined together in a T-shaped geometry. Of particular interest, single ions were shuttled around a 90° corner using control voltage sequences accounting for the complicated electric potential in the junction region. This thesis also details general design strategies which were used to create the non-trivial control voltage sequences used to execute the shuttling protocols.

Acknowledgements

First of all, I would like to thank my research advisor Professor Christopher Monroe for giving me the opportunity to work on this amazing research and also for sharing so much of his wisdom, both scientific and non scientific. Thanks go out to the whole ion trapping group. In particular, special thanks to the whole T-Trap crew, Winfried 'Winni' Hensinger, Prof. James Rabchuk, Steve Olmschenk and fellow undergraduate David Hucul. All of you are not only great colleagues but more importantly great friends.

I would also like to thank all my professors in the Department of Physics and Mathematics for giving me an excellent academic grounding and also to the School of Music who gave me so many opportunities to play and perform with very loud toys.

On a personal note, I would like to thank Goh Teow Lim for all her love and support. I would also like to thank Melvin Leok for his sagely advise.

Finally, I would like to thank my parents for all their hard work and sacrifices which got to me to where I am today.

CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Don't Panic

-Douglas Adams

The research field of quantum computing is a marriage of quantum mechanics and computer science. The field began when Deutsch realized that by taking advantage of quantum phenomenon, certain computations could be done more efficiently than any classical computer [1], [2].

The power of a quantum computer over a classical computer comes from quantum parallelism. Unlike classical bits, which can only be in one state at a time, a quantum bit (qubit) in general exists as a superposition of all possible states. Therefore, a set of n qubits will in general exist as a superposition of 2^n states. A quantum computer takes advantage of this and acts on ALL of these states simultaneously. In contrast, a classical computer can only evolve one set of inputs at a time. Therefore, a quantum computer can potentially exponentially speed up certain calculations.

A "killer application" for quantum computing is Shor's algorithm [3]. Shor's algorithm efficiently factorizes large numbers that are products of two primes. Shor's algorithm is of great importance since modern encryption schemes, most notably RSA, all rely on the computational intractability of factorizing large numbers.

On a more fundamental level, Feynman [4] also showed that a classical computer will in general not be able to efficiently simulate a quantum system. On the other hand, a quantum simulator will be able to efficiently simulate such systems. This is of particular interest in calculating the dynamics of strongly interacting quantum systems, for example, condensed matter systems.

Having considered the motivation to build a quantum computer, we must now think of how to actually build one. The ion trap stands out as a compelling quantum computing architecture [5]. The qubits are well isolated from the environment, the qubits can be measured with near perfect efficiency and the coulomb interaction between qubits allows for the implementation of Universal Quantum Gates [6].

One of the chief challenges for ion trap quantum computing is scalability. Several key quantum algorithms have been demonstrated using a small number of qubits, for example Shor's algorithm [7] and Grover's algorithm [8]. However, these algorithms were implemented with only a small number of qubits (<10). For a quantum computer to do a meaningful calculation, we will need many more ions. For example, to factorize the 193 digit number known as RSA-640, we would need to deterministically control at least 640 qubits.

The first proposal for a quantum computer by Cirac and Zoller [9] proposed having a linear crystal of ions held in a single harmonic trap, however, severe technical difficulties indicate that this scheme may only be good for computations involving only approximately 10 qubits.

To get around these technical problems, Kielpinski et al [10] proposed an ion trap quantum computer architecture consisting of a two dimensional array of ion traps. Control electrodes would then shuttle ions between trapping zones see figure 1.1. In addition, the multi-dimensionality of the ion trap array allows us greater freedom in the computing architecture. For example, in the style of the von Neumann computing architecture we may use some regions of the trap array as our memory whilst another region of the ion trap array may act as the processor.

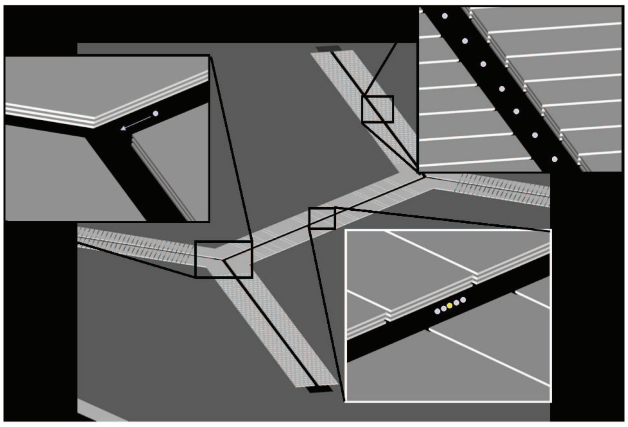

Fig. 1.1: Schematic Representation of a two dimensional ion trap array. The different zones may be specialized to serve certain functions. The magnified box on the right depicts a ion storage region. This will be analogous to the Hard drive of a classical computer. The magnified box in the center depicts an interaction zone. Lasers are used to perform qubit rotations and entanglement operations. This region is analogous to the Central Processing Unit of a Classical Computer.

Though there are several methods by which ions in different trapping regions can interact, the subject of this thesis is the study of physically shuttling the ions around the trapping zones. This is the most direct way to transport information around our quantum computer architecture. We begin by discussing the construction of an ion trap architecture, known as the T-junction ion trap array. Using this ion trap architecture as an example, we describe general strategies to develop key shuttling protocols. Using these strategies, we design and implement key shuttling protocols on the T-trap which culminates in the demonstration of the multi-dimensional arbitrary control of atomic ions.

1.1 The Qubit, ionic Cadmium 111

The Qubit in our ion trap quantum computer is a Cadmium Ion with mass number 111. Like typical ionic species used in quantum computing research, $^{111}Cd^{+}$ is hydrogen like, i.e. there is one valence electron with a $^{2}S_{1/2}$ ground state. In addition, the nucleus has non-zero spin, in the case of Cd 111, $I = 1/2$. This is so that there is a hyperfine splitting in the ground state and it is in these hyperfine levels with which we store our qubit state. $|0\rangle$ is defined to be the state $|F = 0, m_F = 0\rangle$ whilst $|1\rangle = |F = 1, m_F = 0\rangle$. (See fig 1.2). These two levels are chosen since they are insensitive to magnetic fields to first order and as such have long coherence times on the order of seconds.

To initialize the state of the qubit, we optically pump into the $|F = 0, m_F = 0\rangle$ state by applying radiation nearly resonant to the $^{2}S_{1/2}(F = 1) \rightarrow {}^{2}P_{3/2}(F' = 1)$.

To detect the state of the ion, σ_- radiation tuned to the cycling transition $^{2}S_{1/2}(F = 1) \rightarrow {}^{2}P_{3/2}(F' = 2)$ is applied to the ion. If the ion is in the $|1\rangle = |F = 1, m_F = 0\rangle$ state, the ion will be pumped into the $m_F = -1$ state. The ion will then cycle between the $^{2}S_{1/2}(F = 1)$ and the $^{2}P_{3/2}(F' = 2)$, spontaneously

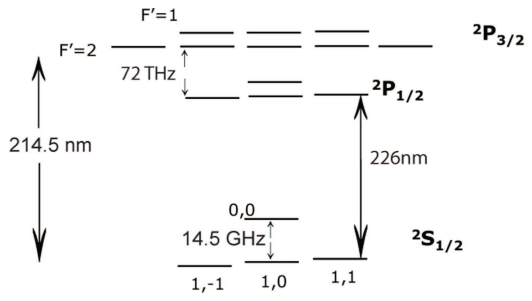

Fig. 1.2: Relevant Energy Levels of the Cadmium 111 ion. Quantum information is stored in the states $|1,0\rangle$ and $|0,0\rangle$

emitting a photon on every cycle. If the ion is in the $|0\rangle = |F = 0, m_F = 0\rangle$, then the radiation will be far detuned from the only available scattering channel, $^2S_{1/2}(F = 0) \leftarrow {}^2P_{3/2}(F' = 1)$ and will thus remain in the $|0\rangle$ state and not scatter any light. In this way, we may efficiently distinguish the two qubit states.

1.1.1 Detection and Doppler Cooling

Consider an ion with two relevant energy levels that are split by some frequency ω_0 and line-width Γ and we apply laser light detuned by frequency $\delta = \nu_{laser} - \nu_{ion}$, then the ion will experience a velocity dependent force given by

$$F = \hbar k \frac{\pi \Gamma s_0}{(1 + s_0 + 4\frac{(\delta - v/\lambda)^2}{\Gamma^2})} \tag{1.1}$$

where k is the wave-number of the laser and $s_0 = I/I_s$ is the on resonance saturation parameter and I_s is the saturation intensity of the transition. By this convention, the positive direction is antiparallel to the wave vector of the laser.

For low ion velocities, we can expand equation 1.1 to first order in v and thus obtain

$$F \approx \hbar k \frac{\pi \Gamma s_0}{(1 + s_0 + 4\frac{\delta^2}{\Gamma^2})} + 8\hbar k \frac{\pi \Gamma \delta s_0}{\Gamma^2 \lambda (1 + s_0 + 4\frac{\delta^2}{\Gamma^2})^2} v \tag{1.2}$$

We first notice that the second term in equation 1.2 is a dissipative force when the de-tuning, δ, is negative, i.e. the beam is red detuned off the resonant frequency, and we may thus cool the atomic ion. In addition, to this dissipative force, there exists a velocity independent force. Unlike typical neutral atom traps, in an ion trap, this force is negligible compared to the electric forces applied to the ion. Thus, we may doppler cool the ion with only one beam. This is unlike a typical neutral atom trap where the velocity component of the force is significant and counter-propagating beams are required to trap the ion.

To doppler cool a Cadmium ion, we apply a beam red detuned off the $^2S_{1/2} \rightarrow ^2 P_{3/2}$ transition. This transition is resonant to 214.5nm light and the line-width (Γ) is approximately 59 MHz. The cooling laser is focused to an approximately $15\mu m$ waist with 1mW of power.

Doppler cooling the ion also presents an additional advantage. As the ion is Doppler Cooled, light is spontaneously emitted in all directions from the ion. Therefore, we use the scattered light from the cooling laser to image the ion.

The Deep UV laser light which we use for detection and cooling is produced by frequency quadrupling the output from a continuous wave Ti:Sapphire laser. The Ti:Sapphire laser is pumped with a Nd:Yag laser at 532 nm and typically produces 1W at 858.0265nm.

To get radiation in the deep UV, we need to double the frequency twice. An LBO crystal inside a build up cavity doubles the frequency from infrared to blue (429nm) and a BBO crystal doubles to the desired deep UV at 214.5nm with typical power of 1mW.

1.2 The Ion Trap

In order to demonstrate the proposed architecture, we constructed a three layer , 49 electrode, 11-trapping zone ion trap array. To demonstrate multidimensional shuttling, the 11-trapping zones are arranged into three linear zones and are connected via a T-junction (see figure 1.3).

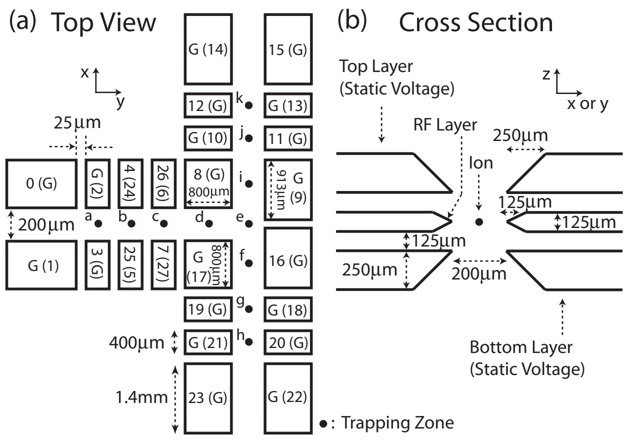

Fig. 1.3: Top and cross sectional schematic of T-junction ion trap array. Dots indicate trapping zones. The 28 control electrodes are numbered with electrodes in the bottom layer in parentheses. Electrodes labeled with G are grounded.

The T-junction ion trap is fabricated with thin laser-machined polished aluminium substrate. The middle layer carries radio-frequency (rf) voltages which provide transverse confinement whilst the top and bottom layers are segmented into independent electrodes. These electrodes carry slowly varying electric potentials and are used to provide axial confinement as well as to implement various shuttling protocols.

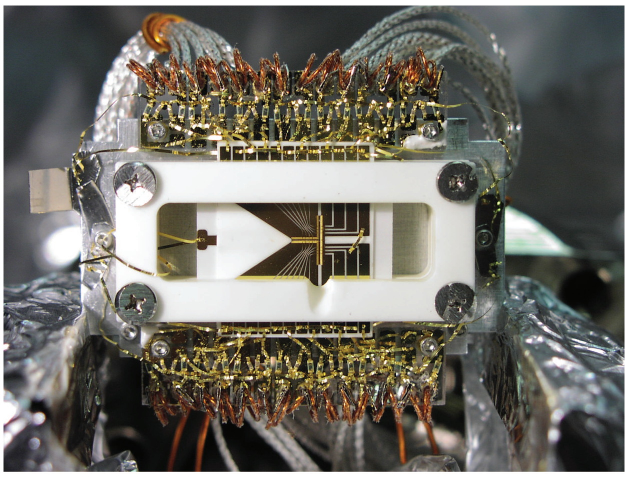

Fig. 1.4: T junction ion trap array prior to being sealed up in vacuum. We note the extensive electrical connections required to provide the voltages to the control electrodes in order to execute shuttling protocols.

The voltages applied to the 28 non grounded control electrodes are provided by analog output cards (NI 6733). These cards can update up to 10^6 times per second. The analog output is amplified using high speed operational amplifiers (Apex, PA85A) which can be slewed at a rate of over 10V per μs. To isolate the control electrodes from external noise, signals pass through a low pass filter consisting of a $1k\Omega$ resistor in series and a $1nF$ capacitor shunted to ground.

Cadmium oxide ovens are heated to provide a cadmium vapor with partial pressure 10^{-11} Torr. Ions are loaded by photoionization of the cadmium vapor with a pulsed laser tuned near the ${}^1S_0 \rightarrow {}^2P_1$ transition for neutral Cadmium

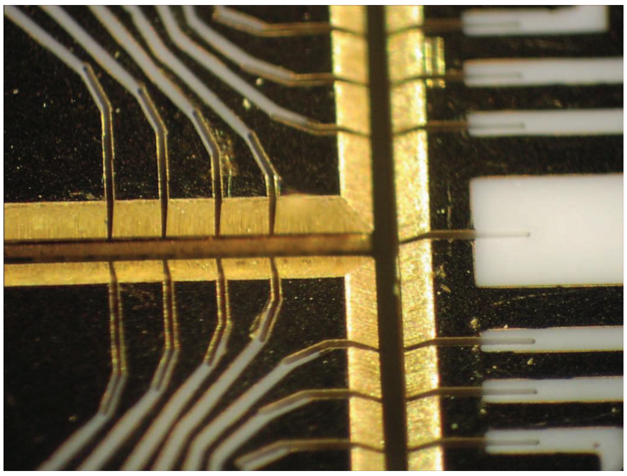

Fig. 1.5: Close up view of T-junction ion trap array showing the junction and all 11 trapping zones. The control electrodes are gold coated via dry film photolithography and wet chemical etching. Electrodes and tracks are formed by depositing 0.015 μ m of titanium followed by 0.4 μm of gold. Two alumina spacer plates are inserted between the two control electrode layers and the central rf electrode layer. All three substrates are held together by rectangular alumina mount bars. Since each layer was fabricated separately, misalignments in the three layers is likely and may be a cause for asymmetries in the shuttling protocols.

[12]. The ions are laser cooled as described in 1.1.1.

Having described the actual construction of the trap, we now consider how the T-trap confines the ions. The following discussion is also relevant to any rf trap geometry.

1.2.1 The rf Paul trap

Earnshaw's theorem shows that due to the divergence free nature of the electric field, electrostatic forces cannot trap a charge [13]. One solution is therefore to use a mixture of radio-frequency (rf) and electrostatic potentials to trap an ion [14]. Such a trap is known as a Paul trap and it is favored for trapped ion quantum computation research. To illustrate the working of a Paul trap, let us consider the T-Trap.

By the principal of linear superposition, we may consider the rf component of the electric field due to the rf electrode independently of the static electric field from the control electrodes and then take the sum of the two contributions. Let the voltage on the surface of the rf electrode to be

$$V_S = V_0 \cos(\omega_T t) \tag{1.3}$$

and set all other surfaces to be ground. The resulting electric potential can be separated into a time dependent component and a space dependent component.

$$V_{rf}(\mathbf{x}, t) = V(\mathbf{x}) \cos(\Omega_T t) \tag{1.4}$$

$V(\mathbf{x})$ is proportional to V_0, therefore, we may define a new function

$$\Psi_{rf}(\mathbf{x}) = V(\mathbf{x})/V_0 \tag{1.5}$$

This implies that

$$V_{rf}(\mathbf{x}, t) = V_0 \Psi_{rf}(\mathbf{x}) \cos(\Omega_T t) \tag{1.6}$$

We calculate the rf component of the electric potential for the T-trap at trapping zone c.

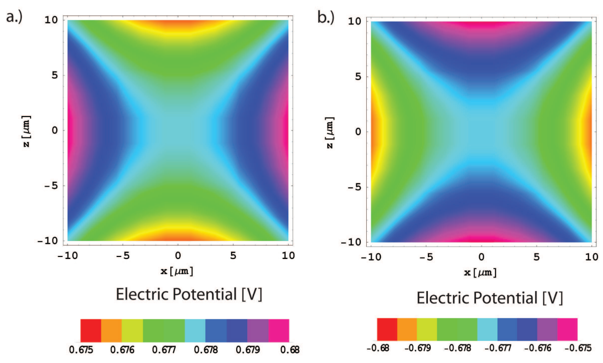

Fig. 1.6: Fig a depicts the electrical potential in the ion trap if 1V is applied to the rf layer. All other electrodes are grounded. As can be seen, the potential is trapping along the x axis and anti-trapping along the z axis. One half period later, the potential on the rf electrode will flip to -1V. The resultant potential is depicted in Fig b. Now, the x axis is anti-trapping while the z-axis is trapping.

From figure 1.6, we see that in both cases, the electric potential is trapping in one direction and anti-trapping in the other. 1/2 a cycle of later, the trapping and anti-trapping axes switch. Thus, the ion on average sees a restoring force in all directions. From Dehmelt, [15], we may approximate equation 1.6 with a pseudo-potential given by

$$\psi(\mathbf{x}) = \frac{e^2}{4m\Omega_T^2}|\nabla V_0 \Psi_{rf}(\mathbf{x})| \tag{1.7}$$

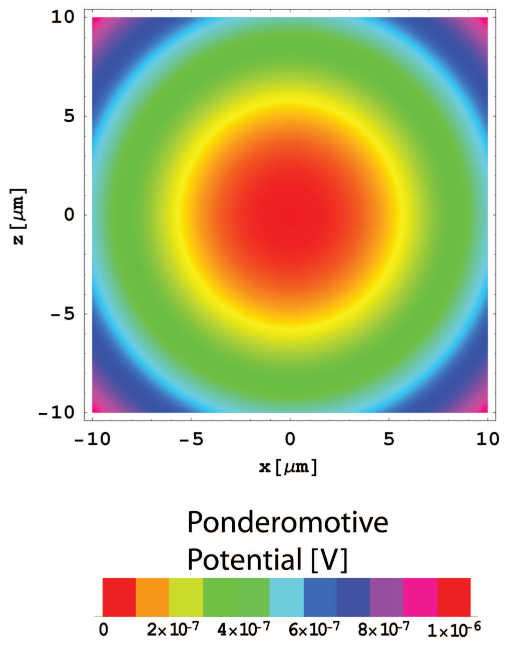

Fig. 1.7: Pseudo-potential as calculated from Equation 1.7. The ion sees a restoring force in both directions. At the middle, the electric fields cancel and thus form an rf node. This is where the ion is trapped.

From Fig 1.7, we verify that the rf pseudo potential does provide trapping in both transverse directions. The pseudo-potential at the minimum is 0 and any point in the trap where the rf pseudo-potential is 0 is denoted an rf node. The node in fig 1.7 extends along the axial direction from trapping zones *a*-*d*.

Similarly, in the other two linear regions, i.e. zones f to h and zones i to k, there also exist similar linear rf nodes.

Near the junction region, the linear rf nodes give way to rf humps. (See figure 1.8) These humps are caused by fringing effects near the junction region and in effect, act as a potential barrier that the ion has to overcome in order to get in and out of the junction region.

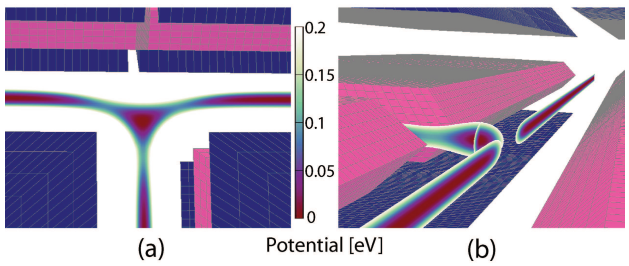

Fig. 1.8: Fig a shows the ponderomotive potential from a view perpendicular to the plane of the trap. Fig b shows a cross-sectional view. Due to fringing effects, the linear rf node cannot extend all the way into the junction region. There exist three linear rf nodes and a point rf node in the middle of the junction. The four nodes are disconnected from each other. Near the junction, there are small humps in the ponderomotive potential. There is where the field due to the rf electrode does not cancel. In the stem of the T, the height of the hump is 0.1eV whilst the height of the two humps at the top of the T are 0.09 eV. The axial width (FWHM) of the humps is approximately 200μm.

Therefore, in this geometry, it is impossible to shuttle the ion from one of the linear regions into the junction region, zone e, without leaving an rf node. This obstacle will be addressed in a later section.

Axial confinement is provided by the control electrodes. To trap an ion at zone c, we need these electrodes to plug the ends of the trap with high voltages on electrodes 8, 17, 4, and 5. If we apply 1 V to the four electrodes and ground all other electrodes, the electric potential in the axial direction is shown in Figure 1.9

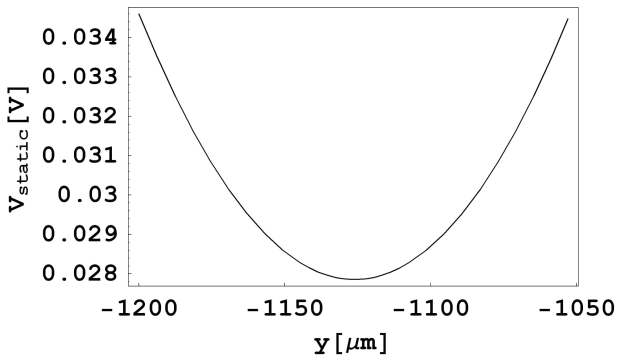

Fig. 1.9: Confinement along the z-axis provided by static potentials on control electrodes. The superposition of the static potential and the ponderomotive potential will therefore confine the ion in all three directions.

We now consider the combination of the rf component and the static component of the electric potentials. We denote $\Psi_i(\mathbf{x})$ to be a basis function of the ith control electrode, where $\Psi_i(\mathbf{x})$ is the solution to the Dirichlet boundary problem specified by applying 1V to the ith control electrode and grounding all other electrodes. By the principal of linear superposition of electric fields, the overall potential will then be

$$\Phi(\mathbf{x},t) = V_0 \cos(\Omega_T t)\Phi_{rf}(\mathbf{x}) + \sum_i V_i \Phi_i(\mathbf{x}) \tag{1.8}$$

and the pseudopotential approximation is given by

$$\psi(\mathbf{x},t) = \frac{e^2}{4\pi\Omega_T^2}|\nabla V_0 \Phi_{rf}(\mathbf{x})|^2 + \sum_i V_i \Phi_i(\mathbf{x}) \tag{1.9}$$

. where V_i is the voltage applied to the ith control electrode.

Due to the complicated nature of ion traps, it is usually difficult or impossible to find analytic solutions for the basis functions and they are therefore obtained numerically via the method of Finite Element Analysis. [16]. Using these basis functions, we designed successful ion shuttling protocols for the T-Trap.

2. CALCULATING ION DYNAMICS

With the calculated basis functions for each electrode of the ion trap, we can then derive the desired potential by suitably superposing the basis functions multiplied by the time varying potential.

$$\Phi(\mathbf{x}, t) = V_0 \cos(\Omega_T t)\Phi_{rf}(\mathbf{x}) + \sum_i V_i(t)\Phi_i(\mathbf{x}) \tag{2.1}$$

Where $\mathbf{x}$ is the three vector denoting position, Ω_T and V_0 are the applied RF frequency and voltage; $V_i(t)$ is the time varying potential applied on the ith electrode and $\mathbf{\Phi_i}(\mathbf{x})$ is the basis function of the ith electrode. Notice here that the multiplier for all the basis functions have explicit time dependence.

The ion's motion due to the electric potential Φ will consist of the low amplitude micro-motion with frequency to the order of Ω_T and the slower but larger amplitude secular motion. Very often, we only need to calculate the secular motion of the ion and ignore the micro-motion, therefore we may approximate Eq 2.4 with a ponderomotive pseudo-potential given by [15]:

$$\psi(\mathbf{x}, t) = \frac{e^2}{4\pi\Omega_T^2}|\nabla V_0\Phi_{rf}(\mathbf{x})|^2 + \sum_i V_i(t)\Phi_i(\mathbf{x}) \tag{2.2}$$

Finally, if there are k ions in the trap, the resultant force on each ion $\mathbf{F_j}$ is given by

$$\mathbf{F_j}(\mathbf{x_1},...,\mathbf{x_n},t) = \begin{cases} -q\nabla\Phi(\mathbf{x_j},t) + \sum_{i\neq j}\frac{q^2}{|\mathbf{x_j}-\mathbf{x_i}|^3}(\mathbf{x_j}-\mathbf{x_i}) & \text{For complete ion motion} \\ -q\nabla\psi(\mathbf{x_j},t) + \sum_{i\neq j}\frac{q^2}{|\mathbf{x_j}-\mathbf{x_i}|^3}(\mathbf{x_j}-\mathbf{x_i}) & \text{For ion secular motion only} \end{cases} \tag{2.3}$$

Therefore, to calculate the dynamics of k ions in a trap we need to solve the set of k coupled second order ordinary differential equations(ODEs):

$$\ddot{\mathbf{x}}_\mathbf{j} = \frac{\mathbf{F_j}}{m}(\mathbf{x_1},...,\mathbf{x_n},t) \equiv \mathbf{a_j}(\mathbf{x_1},...,\mathbf{x_n},t) \tag{2.4}$$

Where j is an integer from 1 to k. Due to the complicated electric field topography, especially in the junction region, it is impossible to solve for the ion's motion analytically, therefore, we need to numerically solve Eq2.4. The design of shuttling protocols require high accuracy solutions of Eq2.4 and as such the numerical evaluation of Eq2.4 is slow. Using a AMD dual core 1.8GHz processor with 2 GB of memory to calculate the trajectory of the ion with a shuttling sequence that brings it from trapping zone d to trapping zone i, the computer time taken to obtain the ion trajectory depends on the ODE solver method and can take anywhere from 5 hours to a week. Therefore, one must make a judicious choice of ODE solver in order to reach the required accuracy in a feasible amount of time.

Extrapolation Class methods and Predictor Corrector Methods are both good choices for efficiently solving ODEs to high accuracy. Extrapolation Class methods are more efficient than Predictor Corrector Methods [19] under most circumstances. However, there are times when evaluating the potential gradient becomes an computationally expensive task. In this case, the Predictor Corrector Methods triumph over the Extrapolation Class methods

A caveat to the two above-mentioned methods is that the calculated electric field has to be smooth. If the electric field is rough, Explicit Runge-Kutta methods are the best choice as these simple methods can "feel" their way around better in treacherous terrain. In addition, if a low accuracy solution is sufficient, single step methods tend to be more efficient than the extrapolation class methods.

Finally, the ODE system Eq.2.4 may be stiff. Stiffness occurs when there are two or more vastly different time scales with which the dependent variables are changing. Many normal ODE solvers are inefficient at numerically evaluating such systems. Fortunately, there are ODE solver methods known as "stiff solvers" that are well suited to handle these systems.

We believe that these four broad classes of ODE solvers should be able to handle almost any ion trap situation. Thus, in the subsequent four sections, we illustrate each of these four classes of methods with a brief discussion of a specific ODE solver.

2.1 Explicit Runge-Kutta Methods

The Explicit Runge-Kutta (ERK) methods are good when only a quick, low accuracy solution is required. In addition, if the potential gradient is rough, the Explicit Runge-Kutta Methods are usually more efficient than other methods.

Explicit Runge-Kutta Methods numerically solve ordinary differential equations (ODEs) of the form

$$\frac{dx}{dt} = f(t, x) \tag{2.5}$$

We let t denote time. The output of any numerical ODE solver is a series of points called nodes. A node is of the form (t_i, x_i) where x_i is an approximation of the exact solution $x(t = t_i)$. The first node is given by the initial conditions. Subsequently every discrete step that the ODE solver takes calculates one more node. Typically, the number of nodes generated is on the order of 10,000.

The most simple version of an ERK method is known as Euler's method. Given a node (t_n, x_n), we wish to calculate, from Euler's method, the next node $(t_{n+1} = t_n + h, x_{n+1})$. This implies that x_{n+1} gives an approximation of the exact value of $x(t = t_n + h)$. The equation to Euler's method is given by

$$x_{n+1} = x_n + hf(x_n, t_n) \tag{2.6}$$

The justification of this method is simple. We assume that the average rate of change of x, $\frac{dx}{dt}$, over the time interval $(t_n, t_{n+1} = t_n + h)$ averages to the rate of change of x at the start of the time interval $\frac{dx(t_n)}{dt}$.

To characterize the error of the Euler method, first consider the Taylor Expansion of x about $t = t_n$.

$$x(t_{n+1} = t_n + h) = x(t_n) + \alpha_1 h + \alpha_2 h^2 + ... \tag{2.7}$$

Where α_i are the Taylor Coefficients. Let us assume that x_n is exact, i.e.

$$x_n = x(t_n) \tag{2.8}$$

Then the numerical estimation x_{n+1}, of $x(t_n + h)$, is given by the equation

$$x_{n+1} = x_n + h\frac{dx}{dt} = x(t_n) + \alpha_1 h \tag{2.9}$$

The error that we get from a single step is thus

$$\epsilon = x(t_n + h) - x_{n+1} = 2\alpha_2 h^2 + 3\alpha_3 h^3 + ... \tag{2.10}$$

. The error of Euler's method is said to be first order since there are no error terms that go by the first power of h.

One problem with Euler's method is that it is asymmetric, i.e. we only use information about the derivative at the start of the time interval. Thus instead of using the derivative at the start of the time interval, we can use the derivative at the middle of the time interval. However, we have no information about what the value of $x(t = \frac{t_n + t_{n+1}}{2})$, i.e. the value of x at the midpoint of the time interval and thus we use Equation 2.82 to take a trial step to estimate $x(t = \frac{t_n + t_{n+1}}{2})$.

$$z_1 = x_n + \frac{h}{2} f(t_n, x_n) \tag{2.11}$$

We then use this intermediate point to calculate the rate of change and thus estimate the next node.

$$x_{n+1} = x_n + \frac{h}{2} f(t_n + h/2, z_1) \tag{2.12}$$

By using the Taylor expansion of x, we find that the leading term of the error function goes by h^3. This method is thus said to be second order accurate and is called the Second Order Runge-Kutta Method (RK2). It is also known as the Midpoint Method.

We can continue to derive other similar methods that have even higher order accuracy. A general prescription to find these methods can be obtained in Leader [17]. However, by far the most commonly used method is the classical fourth order Runge-Kutta Method (RK4). The equation of this method is given by

$$x_{i+1} = x_n + \frac{h}{3}(\frac{g_1}{2} + g_2 + g_3 + \frac{g_4}{2}) \tag{2.13}$$

Where the values of g_m are given by

$$g_1 = f(t_n, x_n) \tag{2.14}$$

$$g_2 = f(t_n + \frac{h}{2}, x_n + \frac{g_1}{2}) \tag{2.15}$$

$$g_3 = f(t_n + \frac{h}{2}, x_n + \frac{g_2}{2}) \tag{2.16}$$

$$g_4 = f(t_n + h, x_n + g_3) \tag{2.17}$$

It turns out that this method is 4th order accurate. I.e. the leading term in the error goes by h^5.

We note that in each step, RK4 does 4 evaluations of the function f and it is 4th order accurate. Similarly, the Euler method does 1 evaluation of f per step and is 1st order accurate and RK2 does 2 evaluations of f and is 2nd order accurate. However, for higher order ERK methods, it turns out that the number of evaluations of f is higher than the order of accuracy. Thus ERK methods with order greater than 4 are considered computationally inefficient since the evaluation of f is considered to be computationally expensive and we always try to minimize the number of evaluations of f during our calculation. This thus explains the popularity of RK4. RK4 has a good tradeoff in terms of accuracy and the computational efficiency since it is the highest order ERK method in which the accuracy order is the same as the number of evaluations of f per step.

Having established RK4 as the most useful of the ERK methods, we proceed to show how to implement RK4 to solve Equation 2.4.

Eq.2.4 is a second order ODE and in order for us to use the ERK method (and other ODE methods for that matter) we need to reformulate Eq.2.4 into a system of 2k first order ODEs where k is the number of ions in our system.

$$\frac{d\mathbf{x_j}}{dt} = \mathbf{v_j} \tag{2.18}$$

$$\frac{d\mathbf{v_j}}{dt} = \mathbf{a}(\mathbf{x_1}, ..., \mathbf{x_n}, t) \tag{2.19}$$

To simplify the notation, we define the following 3k dimensional vectors.

$$\mathbf{X}(t) = \begin{pmatrix} \mathbf{x}_1(t) \\ \vdots \\ \mathbf{x}_k(t) \end{pmatrix} \tag{2.20}$$

$$\mathbf{V}(t) = \begin{pmatrix} \mathbf{v}_1(t) \\ \vdots \\ \mathbf{v}_k(t) \end{pmatrix} \tag{2.21}$$

$$\mathbf{A}(\mathbf{x},t) = \begin{pmatrix} \mathbf{a}_1(\mathbf{x},t) \\ \vdots \\ \mathbf{a}_k(\mathbf{x},t) \end{pmatrix} \tag{2.22}$$

Thus, **X** denote the positions of the k ions, **V** denote the velocities of the k ions and **A** denote the accelerations of the k ions.

Starting with the node $(t_n, \mathbf{X_n}, \mathbf{V_n})$ we wish to calculate the next node $(t_{n+1} = t_n + h, \mathbf{X_{n+1}}, \mathbf{V_{n+1}})$. Thus we will calculate $\mathbf{X_{n+1}}$ which will approximate the exact solution $\mathbf{X}(t_n + h)$ and $\mathbf{V_{n+1}}$ which will approximate $\mathbf{V}(t_n + h)$.

The equations to do so are similar to Equation 2.13 with the exception that we must replaces scalar quantities with vector quantities.

$$\begin{pmatrix} \mathbf{V_{n+1}} \\ \mathbf{X_{n+1}} \end{pmatrix} = \begin{pmatrix} \mathbf{V_n} \\ \mathbf{X_n} \end{pmatrix} + \frac{h}{3}(\frac{1}{2}\begin{pmatrix} \mathbf{g_1} \\ \mathbf{f_1} \end{pmatrix} + \begin{pmatrix} \mathbf{g_2} \\ \mathbf{f_2} \end{pmatrix} + \begin{pmatrix} \mathbf{g_3} \\ \mathbf{f_3} \end{pmatrix} + \frac{1}{2}\begin{pmatrix} \mathbf{g_4} \\ \mathbf{f_4} \end{pmatrix}) \tag{2.23}$$

Where

$$\begin{pmatrix} \mathbf{g_1} \\ \mathbf{f_1} \end{pmatrix} = \begin{pmatrix} \mathbf{A}(\mathbf{X_n}, t_n) \\ \mathbf{V_n} \end{pmatrix} \tag{2.24}$$

$$\begin{pmatrix} \mathbf{g_2} \\ \mathbf{f_2} \end{pmatrix} = \begin{pmatrix} \mathbf{A}(\mathbf{X_n} + \frac{1}{2}h\mathbf{f_1}, t_n + \frac{h}{2}) \\ \mathbf{V_n} + \frac{h}{2}\mathbf{g_1} \end{pmatrix} \tag{2.25}$$

$$\begin{pmatrix} \mathbf{g_3} \\ \mathbf{f_3} \end{pmatrix} = \begin{pmatrix} \mathbf{A}(\mathbf{X_n} + \frac{1}{2}h\mathbf{f_2}, t_n + \frac{h}{2}) \\ \mathbf{V_n} + \frac{h}{2}\mathbf{g_2} \end{pmatrix} \tag{2.26}$$

$$\begin{pmatrix} \mathbf{g_4} \\ \mathbf{f_4} \end{pmatrix} = \begin{pmatrix} \mathbf{A}(\mathbf{X_n} + h\mathbf{f_3}, t_n + h) \\ \mathbf{V_n} + h\mathbf{g_3} \end{pmatrix} \tag{2.27}$$

An adaptive step size algorithm is almost always used in conjunction with the Runge-Kutta Method in order to maximize computational efficiency. After every step h is taken, we estimate the error of the nodal value $\mathbf{V_{n+1}}$, $\mathbf{X_{n+1}}$. The way to estimate the error is via the method of RK pairs[17]. For example, we simultaneously use the 4th order Runge-Kutta method and a 5th order Runge-Kutta method to derive $\mathbf{V_n}$, $\mathbf{X_n}$. We then compute the difference of the two evaluations at each step. If the difference is sufficiently small, i.e. such that it is within our error goals, we move on to evaluate the next node, otherwise, we reduce the step size and repeat the procedure till the error is sufficiently small. This algorithm is implemented in $Matlab$ with the function ode45. We could just as well have implemented this system with the second order ERK method and the third order ERK method. This can also be implemented in Matlab with the function ODE23. In addition, more complicated versions of the Explicit Runge-Kutta solver in addition to the adaptive step size algorithm also adaptively alter the order of the Runge-Kutta solver method. Thus, one step may use the RK pair, RK4 and RK5, whilst the next step may use RK2 and RK3. Such a algorithm can be called in Mathematica with the function NDSolve and by specifying the ODE solver to be ExplitRungeKutta.

2.2 Bulirsch-Stoer Method

The Bulirsch-Stoer Method is an ODE solver that yields high accuracy solutions efficiently[18]. However, it is inefficient at yielding low accuracy solutions and would be a poor choice if a "quick and dirty" solution is desired[19]. In addition, if the forces on the particle are rough or discontinuous, this method is not effective. Instead, the Runge-Kutta methods would be a better choice. However, we found that almost all the time, the Bulirsch-Stoer method was the most efficient ODE solver method and we used as this method as our workhorse solver. Thus, we will discuss this method in somewhat greater detail.

From Equation 2.9, we see that the result of such a numerical calculation is a polynomial function of h. If $h = 0$ we would then get the exact answer. However, if h were to be 0, we would end up taking an infinitely long time to solve our ODE. However, if we were to probe Equation 2.9 by running our ODE solver with different values of h, we would find several points of Equation 2.9. We could then fit a polynomial through these points and read off the value of the polynomial at the magical point $h = 0$ to get a considerably more accurate value. This idea is known as Richardson Extrapolation.

Therefore the Bulirsch-Stoer Method goes as such.

1.) Use some ODE solver and run it with different sub-step sizes h to generate several preliminary estimates.

2.) Apply Richardson Extrapolation to these preliminary estimates and gain a hugely improved estimate.

Thus to begin our discussion of the Bulirsch-Stoer method we will first discuss polynomial interpolation. Given a set of n points.

$$\{(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)\} \tag{2.28}$$

There is a unique n-1 order polynomial that passes through each of the n points. For example, with two points, there is a linear polynomial, for three points, there is a unique quadratic, etc. This polynomial is given by Lagrange's Classical Formula

$$\begin{aligned} P(x) &= \frac{(x-a_2)(x-a_3)...(x-a_n)}{(a_1-a_2)(a_1-a_3)...(a_1-a_n)}b_1 + \frac{(x-a_1)(x-a_3)...(x-a_n)}{(a_2-a_1)(a_2-a_3)...(a_2-a_n)}b_2 + ... \\ &+ \frac{(x-a_1)(x-a_2)...(x-a_{n-1})}{(a_n-a_1)(a_n-a_3)...(a_n-a_{n-1})}b_n \end{aligned} \tag{2.29}$$

Although we can evaluate this polynomial in full by Equation 2.29, in the context of Richardson Extrapolation, we only are interested in a single value of the polynomial, i.e. at $x = 0$. In addition, Lagrange's formula is unsatisfactory as it provides no error estimate.

Neville's Algorithm [20] is a much better algorithm for our purposes. Given our set of n points, Neville's Algorithm finds the one value of $P(x)$ (defined in Equation 2.29) that we are interested in without finding all of the unnecessary coefficients of $P(x)$. In addition, Neville'e algorithm also provides a simple error estimate.

To illustrate this algorithm, we will fit the following three points to a second order polynomial and find the value of that polynomial at $x = 0$. The three points are

$$\{0.0225, 1.34838), (5.625 * 10^{-3}, 1.34948), (2.5 * 10^{-3}, 1.34969)\} \tag{2.30}$$

These numbers are derived from solving an ODE system with different sub-step sizes. We will derive these three points later in this section.

We define $P_{(1)}$ to be the value at $x = 0$ of the unique zero order polynomial passing through the first point. Thus, $P_{(1)} = 1.34838$. Similarly, $P_{(2)}$ will be the second ordinate of the second point, i.e. $P_2 = 1.34948$ and $P_3 = 1.34969$. Next we define $P_{(1)(2)}$ to be the value of the unique first order polynomial that passes through the first two points at $x = 0$. This value can be derived from P_1 and P_2 by the equation.

$$P_{(i)(i+1)...(i+m)} = \frac{-x_{i+m}P_{i(i+1)...(i+m-1)} + x_i P_{(i+1)(i+2)...(i+m)}}{x_i - x_{i+m}} \tag{2.31}$$

From Equation 2.31 we can also derive the value of P_{23}, the value of the first order polynomial that passes through the second and third point. Our final value, P_{123} is the value of the unique second order polynomial that passes through all three points at $x = 0$. Again, P_{123} can be derived from P_{12} and P_{23} via equation 2.31.

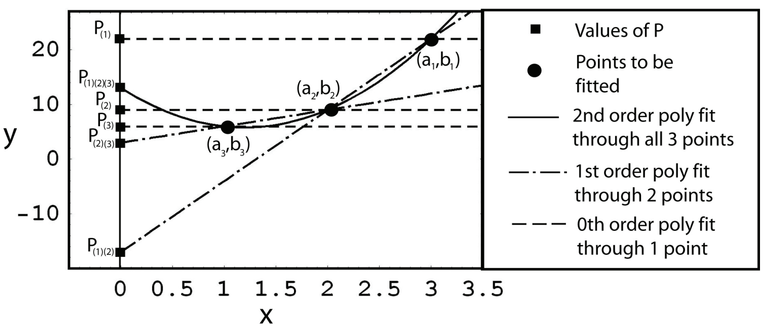

Fig. 2.1: We demonstrate Neville's algorithm for three random points $\{(a_1, b_1), (a_2, b_2), (a_3, b_3)\}$. The y intercept of the six polynomials give us the various P's. Finally, Neville's algorithm does not actually calculate the interpolating polynomials but only the y intercepts of the polynomials. The interpolating polynomials are added for illustrative purposes.

We can think of the various values of P to form a triangular table (also known as a "tableau" in the literature.)

$$\begin{array}{ccccccc} P_1 & & & & 1.34838 & & \\ & P_{12} & & & & 1.34985 & \\ P_2 & & P_{123} & = & 1.34948 & & 1.34986 \\ & P_{23} & & & & 1.34948 & \\ P_3 & & & & 1.34969 & & \end{array} \tag{2.32}$$

For any P, if there is a column on the left, then the two adjacent P's are the parents and the P that we started with is the daughter element. Using Equation 2.31, we can thus fill in this triangular table column by column from left to right. The Right Hand Side of Equation 2.32 shows us the values that we derived by applying 2.31. To estimate the error of our result, we can use the difference between the parent values and the daughter value to gain a crude estimate the error. For example, the error estimate of $P_{(1)(2)(3)}$ is given by

$$\epsilon = max\{|P_{123} - P_{12}|, |P_{123} - P_{23}|\} = 8.6 * 10^{-6} \tag{2.33}$$

One last point to note about Neville's algorithm is that if we need to fit a new point into our polynomial, we need not recalculate all the values of P from scratch. For example, if the points that we need to fit are instead

$$\{0.0225, 1.34838), (5.625 * 10^{-3}, 1.34948), (2.5 * 10^{-3}, 1.34969), (a_4, b_4)\} \tag{2.34}$$

Then all we need to calculate are $P_{(4)}$, $P_{(3)(4)}$, $P_{(2)(3)(4)}$ and $P_{(1)(2)(3)(4)}$ where

$P_{(1)(2)(3)(4)}$ is now the final answer that we desire.

With the machinery of polynomial interpolation in our grasp, we now need to generate the raw numerical estimates that we will feed into Neville's algorithm. The Bulirsch-Stoer method solves ODEs of the form of Equation 2.5. Each Bulirsch-Stoer step is a large step in time and we denote it to be H. Thus with each Bulirsch-Stoer step, we begin with a node (t_i, x_i) and generate the next node $(t_{i+1} = t_i + H, x_{i+1})$.

The ODE solver that we use to generate our initial rough estimates of $x(t_i + H)$ is known as the Modified Midpoint Method. The modified midpoint method advances across the time interval (t_i, t_{i+1}) in a series of n uniform sub-steps of size $h = H/n$. The equations for this method are

$$z_0 = x_i \tag{2.35}$$

$$z_1 = z_0 + hf(t_i, z_0) \tag{2.36}$$

$$z_{m+1} = z_{m-1} + 2hf(t_i + mh, z_m) \tag{2.37}$$

$$\chi_{i+1}(h = H/n) = \frac{1}{2}[z_n + z_{n-1} + hf(t_i + H, z_n)] \tag{2.38}$$

Every sub-step of the modified midpoint method, calculates a value z_m. These values are intermediate calculations and once we find the value $\chi_{i+1}(h = H/n)$, the various z's are immediately discarded.

$\chi_{i+1}(H/n)$ are our initial rough estimates of the exact value $x(t = t_{i+1})$, with this notation, we intend to emphasize the assumption that the result of this numerical calculation is a function of the Modified Midpoint Method sub-step size h. Thus, the various values of χ_{i+1} will be fed into Neville's Algorithm.

Equation 2.37 is very similar to the Midpoint Method Equation 2.12. In order to advance from the node $(t_i + mh, z_m)$ to the point $(t_i + (m+2)h, z_{m+2})$, we use information from the derivative at the middle of the time interval $(t_i + mh, t_i + (m+2)h)$ to calculate the next point, i.e. $f(t_i + (m+1)h, z_{m+1})$. The main difference between the two is that the Modified Midpoint Method uses two previously calculated nodes to evaluate the next node whereas the Midpoint method uses only one previous node. This implies that the Modified Midpoint Method is more computationally efficient than the Midpoint Method since in every step of the Modified Midpoint Method, there is only 1 evaluation of f whereas for the Midpoint Method, there are 2 evaluations of f per step. This superior computational efficiency is one reason why the Modified Midpoint Method is used as our base ODE solver.

The second reason for our choice of the Modified Midpoint Method is because the form of χ_{i+1} contains only even powers of h, [19] i.e.

$$\chi_{i+1}(H/n) = x(t_i + H) + \sum_{j=1}^{\infty}(c_j(\frac{H}{n})^{2j}) \tag{2.39}$$

. This is a good thing for two reasons.

1.) Newton's equations are time reversible and thus we thus it is desirable if our ODE solver is also time reversible [21]. The form of the error function implies that the Modified Midpoint Method is time reversible[21]. To see why the this is true, we notice that Equation 2.39 is even about $h = 0$.

$$\chi_{i+1}(-H/n) = x(t_i + H) + \sum_{j=1}^{\infty}(c_j(\frac{H}{n})^{2j}) \tag{2.40}$$

. A negative sub-step size $-H/n$ just means that we are numerically estimating $(x(t = t_i))$, we denote this numerical estimate as y, from the node $(t_i + H, \chi_{i+1}(H/n))$ by running the Modified Midpoint Method backwards in time. Thus equation 2.40 means that in order for y to be equal to x_i, then $\chi(-H/n)$ must take the value of Equation 2.39. This thus implies that if we run the Modified Midpoint Method forward such that we derive $\chi_{i+1}(H/n)$ from (x_n); then, if we run the Modified Midpoint Method backwards such that we derive the value y from $\chi_{i+1}(H/n)$, then $y = x_n$. Thus we show that the Modified Midpoint Method is time reversible.

2.) From equation 2.39 we see that χ_{i+1} is both a power series of h and a power series of h^2. Say we have calculated $\chi_{i+1}(h = H/2)$ and $\chi_{i+1}(h = H/4)$, we will then have two choices of how to do Richardson Extrapolation to estimate the value $\chi(h = 0) = x(t_{n+1})$. Firstly, we may either use the fact that χ_{i+1} is a power series of h and thus apply Neville's algorithm, Equation 2.31, on the points

$$\{(h = H/2, \chi_{i+1}(h = H/2)), (h = H/4, \chi_{i+1}(h = H/4))\} \tag{2.41}$$

to obtain our estimate of $\chi(h = 0)$.

Secondly, we may use the fact that χ is a power series of h^2. In this case we would then apply Neville's algorithm to the points

$$\{(h^2 = H^2/2^2, \chi(h^2 = H^2/2^2)), (h^2 = H^2/4^2, \chi(h^2 = H^2/4^2))\} \tag{2.42}$$

to obtain our estimate of $\chi(h^2 = 0)$. Note here that the values $\chi(h = H/n) = \chi(h^2 = H^2/n^2)$.

It turns out that fitting the latter pair of points, Eq. 2.42 yields a much better numerical result. In order to see why this is true, we consider an analogous but more concrete example.

The function

$$\cos(x) = 1 - x^2/2! + x^4/4! - ... \tag{2.43}$$

is both a power series of x and x^2. Let us use Neville's algorithm to estimate the value $\cos(0)$ from the values $\cos(0.1)$ and $\cos(0.2)$. Using the fact that $\cos(x)$ is a power series of x, we fit the points

$$\{(0.1, \cos(x = 0.1)), (0.2, \cos(x = 0.2))\} \tag{2.44}$$

with Neville's algorithm. The estimation of $\cos(0)$ we obtain is $1.00994(6\ significant\ figures)$. The fractional difference of this numerical estimate from the exact solution is 10^{-2}.

However, if we use the fact that $\cos(x)$ is a power series of x^2, we then fit the points

$$\{(0.1^2, \cos(0.1)), (0.2^2, \cos(0.2))\} \tag{2.45}$$

with Neville's algorithm. The estimation of $\cos(0)$ we now obtain is 0.999983

(6 significant figures). The fractional difference of this numerical estimate from the exact solution is 10^{-5}. Thus, the latter extrapolation is 1000 times more accurate than the former.

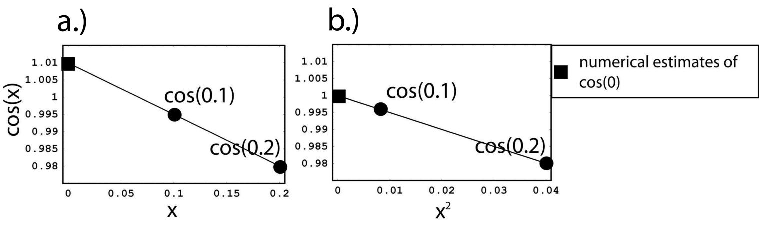

Fig. 2.2: Fig a plots $\cos(x)$ against x whilst fig b plots $\cos(x)$ against x^2. The numerical estimate from Fig b is much closer to the exact value, 1, than the numerical estimate from fig a.

Fig 2.2 a shows the result Neville's algorithm applied to the first pair (Eq.2.44) of points and Fig 2.2 b shows Neville's algorithm applied to the second pair of points (Eq.2.45). From Fig 2.2, we can see why the latter method is superior. cos(0.1) is a better approximation of cos(0) than cos(0.2). In Fig 2.2 b, the point representing cos(0.1) is 4 times closer to the vertical axis than the point representing cos(0.2). In Fig 2.2 a, the point representing cos(0.1) is only 2 times closer to the vertical axis than the point representing cos(0.2). Therefore, the numerical estimate derived from the second pair of points, derived from the fact that *cosine* is a power series of h^2, depends much more strongly on the value cos(0.1) than the numerical estimate derived from the first pair of points. Thus the superiority of the second method is expected.

Similarly, $\chi(h = H/4)$ is a superior numerical calculation compared to $\chi(h = H/2)$. By the same reasoning as our example above, if we fit the points given by 2.42, the final numerical approximation of $\chi(h = 0)$ will depend much more strongly on the value $\chi(h = H/4)$ than if had fit the points given by 2.44. Thus, by choosing to use the fact that χ is a power series in h^2 we gain much more accurate than if we had used the fact that χ is a power series in h.

With all the tools that we need in hand, we now illustrate the Bulirsch-Stoer method by numerically solving a simple differential equation.

$$\frac{dx}{dt} = x \tag{2.46}$$

$$x(t = 0) = 1 \tag{2.47}$$

Thus the right hand side of equation 2.5 is

$$f(t, x) = x \tag{2.48}$$

The exact solution to equation 2.46 is

$$x = e^t \tag{2.49}$$

For our example, we will take one large Bulirsch-Stoer step from $t = 0$ to $t = 0.3$. Recall that H denotes the size of the Bulirsch-Stoer step and thus $H = 0.3$. The result of this calculation will thus be a numerical estimate of the exact solution $x(t = 0.3)$. We note that since the first node is given by initial conditions, $(t_0 = 0, x_0 = 1)$, we are trying to calculate the value $x_1 \approx x(t = 0.3)$.

1.) We being by finding rough estimates of $x(t = 0.3)$ by the Modified Midpoint Method with 2 sub-steps and 4 sub-steps.

2.) Apply Richardson Extrapolation to available points and estimate the error.

3.) If we are within our error goals, record the result of step 2 and go on to calculate the next node. Otherwise, repeat step 2 with an additional point derived from a higher number of step sizes.

The number of sub-steps is given by the sequence

$$n = 2, 4, 6, 8, 10, 12, 14, 16, ... \tag{2.50}$$

However, if the solution does not meet our error goals beyond a certain value of n, this would indicate that there is some obstacle in the time interval of the Bulirsch-Stoer step. Therefore, this sequence is usually terminated at the 8th iteration which corresponds to n=16 sub-steps. Upon which, instead of subdividing the interval indefinitely, H is reduced (usually halved) and we will repeat the above procedure. A discussion of the adaptive step size algorithm is provided in Numerical Recipes.[19]

Following our 3 step procedure which was outlined above, we first find $\chi_1(h^2 = H^2/2^2)$ and $\chi_1(h^2 = H^2/4^2)$. The values we obtain to six significant figures are

$$\begin{aligned} \chi_1(h^2 = H^2/2^2) &= 1.34838 \qquad (2.51) \\ \chi_1(h^2 = H^2/4^2) &= 1.34948 \end{aligned}$$

Notice that these are the same values as $P_{(1)}$ and $P_{(2)}$ in Equation 2.32. Applying

Richardson Extrapolation to these points, the value of x_1 we obtain is

$$x_1 = P_{(1)(2)} = 1.34985(to\ six\ significant\ figures) \tag{2.52}$$

From Equation 2.49, $x(t = 0.3) = e^{0.3} = 1.34986(6\ significant\ figures)$. The fractional difference of $\chi_1(h^2 = H^2/2^2)$ from the exact solution is $1.1 * 10^{-3}$, whilst the fraction difference of $\chi_1(h^2 = H^2/4^2)$ from the exact solution is $2.8 * 10^{-4}$. However, by applying Richardson Extrapolation to these two points, the new refined value has a fractional difference of $6.4 * 10^{-6}$. Thus, the Richardson yields a result that is 40 times more accurate than our raw estimates.

In general, we should not know the exact error of our ODE, otherwise, there would be no reason to numerically calculate a solution. Therefore we can use our error estimation scheme Eq.2.33. With this we estimate the fractional error to be $1.5 * 10^{-3}$. This is higher than the actual error.

If a estimated fractional error of $1.5 * 10^{-3}$ is not acceptable, we then proceed to find the value of $\chi(H/6)$.

$$\chi_1(H^2/6^2) = P_{(3)} = 1.34969(to\ six\ significant\ figures) \tag{2.53}$$

$\chi_1(H^2/6^2)$ has a fractional error of $1.1 * 10^{-4}$. Notice here that the Richardson Extrapolation from the two points $\{\chi_1(h^2 = H^2/2^2), \chi_1(h^2 = H^2/4^2)\}$, is already 20 times more accurate than $\chi_1(H^2/6^2)$.

However, if we are to apply Richardson Extrapolation to all three points, we obtain a value of 1.34986 ($6\ significant\ figures$). This has an exact fractional error of $1.5 * 10^{-8}$. This numerical result is thus about 10,000 times more accurate than our best χ_1 value, $\chi_1(H^2/6^2$.

The error estimate is given by Equation 2.33 and is $8.6 * 10^{-6}$. Again this error estimate is considerably higher than the true error.

To summarize the above numerical results

Substepsize	Fract. error from Mod. Midpoint	Fract. Error after Richardson Extrap.
$h = H/2$	$1.1 * 10^{-3}$	NA
$h = H/4$	$2.8 * 10^{-4}$	$6.4 * 10^{-6}$
$h = H/6$	$1.1 * 10^{-4}$	$1.5 * 10^{-8}$

Tab. 2.1: The first column refers to the fractional difference obtained in the numerical estimation of $x(t = 0.3)$ and the exact solution. The second column shows the fractional difference after Richardson Extrapolation is applied. Notice that Richardson Extrapolation provides a huge increase in accuracy

Finally, we generalize the Bulirsch-Method to solve Equation 2.4.

Given a node $(t_i, \mathbf{V_i}, \mathbf{X_i})$, we would like to calculate the next node $(t_{i+1} = t_i + H, \mathbf{V_{i+1}}, \mathbf{X_{i+1}})$. Therefore, $\mathbf{V_{i+1}}$ approximates the exact velocities $\mathbf{V}(t_i + H)$ and $\mathbf{X_{i+1}}$ approximates the exact ion positions $\mathbf{X}(t_i + H)$

The modified midpoint method crosses the time interval $t_{n+1} = t_{n+1} + H$ through a sequence of n uniform sub-steps in time with uniform sub-step size.

$$h = \frac{H}{n} \tag{2.54}$$

The formulae of the method to calculate the motion for the jth ion are given by

$$\begin{pmatrix} \mathbf{y_0} \\ \mathbf{z_0} \end{pmatrix} = \begin{pmatrix} \mathbf{V}(t_i) \\ \mathbf{X}(t_i) \end{pmatrix} \tag{2.55}$$

$$\begin{pmatrix} \mathbf{y_1} \\ \mathbf{z_1} \end{pmatrix} = \begin{pmatrix} \mathbf{y_0} \\ \mathbf{z_0} \end{pmatrix} + h \begin{pmatrix} \mathbf{A}(\mathbf{z_0}, t_i) \\ \mathbf{y_0} \end{pmatrix} \tag{2.56}$$

$$\begin{pmatrix} \mathbf{y_{m+1}} \\ \mathbf{z_{m+1}} \end{pmatrix} = \begin{pmatrix} \mathbf{y_{m-1}} \\ \mathbf{z_{m-1}} \end{pmatrix} + 2h \begin{pmatrix} \mathbf{A}(\mathbf{z_m}, t_i + mh) \\ \mathbf{y_m} \end{pmatrix} \tag{2.57}$$

$$\begin{pmatrix} \tilde{\mathbf{X}}_{\mathbf{i+1}}(H^2/n^2) \\ \tilde{\mathbf{V}}_{\mathbf{i+1}}(H^2/n^2) \end{pmatrix} = \frac{1}{2}[\begin{pmatrix} \mathbf{y_n} \\ \mathbf{z_n} \end{pmatrix} + \begin{pmatrix} \mathbf{y_{n-1}} \\ \mathbf{z_{n-1}} \end{pmatrix} + h \begin{pmatrix} \mathbf{A}(\mathbf{z_n}, t_i + H) \\ \mathbf{y_n} \end{pmatrix}] \tag{2.58}$$

$\mathbf{A}$ refers to the acceleration of the k ions. In addition, $\mathbf{z_i}$, $\mathbf{y_i}$ are intermediate vectors that the modified midpoint method calculate at each sub-step. Once the calculation of each $\tilde{\mathbf{X}}_{\mathbf{i+1}}(H^2/n^2)$ and $\tilde{\mathbf{V}}_{\mathbf{i+1}}(H^2/n^2)$ is complete, the values $\mathbf{z_i}$, $\mathbf{y_i}$ are immediately discarded.

$\tilde{\mathbf{X}}_{\mathbf{i+1}}(H^2/n^2)$ is the desired approximation to the exact solution $\mathbf{X}(t = t_i{+}H)$ and $\tilde{\mathbf{V}}_{\mathbf{i+1}}(H^2/n^2)$ is the desired approximation to the exact solution $\mathbf{V}(t = t_i + H)$. Again, we emphasize that $\tilde{\mathbf{X}}_{\mathbf{i+1}}$ and $\tilde{\mathbf{V}}_{\mathbf{i+1}}$ are functions of h^2 where $h = H/n$ is the Modified Midpoint Method substep size.

In order to do implement the Bulirsch-Stoer method, we follow the same procedure as outlined above.

1.) We being by finding rough estimates of the function $\mathbf{X}(t_n + H)$ and $\mathbf{V}(t_n + H)$ by the Modified Midpoint Method with 2 sub-steps and 4 sub-steps.

2.) Apply Richardson Extrapolation to available points and estimate the error.

3.) If we are within our error goals, record the result of step 2 and go on to calculate the next node. Otherwise, repeat step 2 with an additional point derived from a higher number of step sizes.

The number of sub-steps to use is given by Equation 2.50. Again, if we do not meet our error goals by n=16, we stop the process and halve the Bulirsch-Stoer step H and repeat the process.

Polynomial interpolation is carried out using Neville's Algorithm (Equation 2.31). However, we note that the each of the various P's in the tableau will now be a $6k$ dimensional vector representing intermediate estimates of the positions and velocities. The $6k$ come from the 6 degrees of freedom of each of the k ions. If we are to apply Richardson's Extrapolation to the three rough estimates obtained by taking n=2, n=4 and n=6, the points that we will interpolate will now be

$$\{(H^2/2^2, \begin{pmatrix} \mathbf{V_2} \\ \mathbf{X_2} \end{pmatrix}), (H^2/4^2, \begin{pmatrix} \mathbf{V_4} \\ \mathbf{X_4} \end{pmatrix}), (H^2/6^2, \begin{pmatrix} \mathbf{V_6} \\ \mathbf{X_6} \end{pmatrix})\} \tag{2.59}$$

and we will initialize the tableau

$$\mathbf{P_1} = \begin{pmatrix} \mathbf{V_2} \\ \mathbf{X_2} \end{pmatrix}, \mathbf{P_2} = \begin{pmatrix} \mathbf{V_4} \\ \mathbf{X_4} \end{pmatrix}, \mathbf{P_3} = \begin{pmatrix} \mathbf{V_6} \\ \mathbf{X_6} \end{pmatrix}, \tag{2.60}$$

Using Equation 2.31 recursively, we can then obtain the final vector $\mathbf{P_{123}}$. In order to estimate the error of $\mathbf{X}$, we can use the difference between norm of the $\mathbf{X}$ component of the parent values and the $\mathbf{X}$ component of the daughter value to gain a crude estimate of the error

$$\epsilon = max\{|\mathbf{P_{123X}} - \mathbf{P_{12X}}|, |\mathbf{P_{123X}} - \mathbf{P_{23X}}|\} \tag{2.61}$$

where $\mathbf{P_{123X}}$ refer to the first j components of the vector $\mathbf{P_{123}}$. Similarly, we can estimate the error in the velocity.

Finally, to give a sense of how this ODE solver operates, the average size of each Bulirsch-Stoer step (H) is on the order of $1ns$ and the evaluation takes 60,000 Bulirsch-Stoer steps. The range of values of H used in the evaluation is approximately $10^{-15}s$ to $10^{-9}s$.

2.3 Adams Predictor Corrector Method

The Adams Predictor Corrector Methods derive from two powerful ideas, the first idea is to combine two ODE solver sub-methods whose complementary properties work together to increase efficiency and accuracy. The second idea is from the so called multi-step or Adams ODE solver methods that use information from previous evaluations to advance the dependent variable. [22]

Two ODE base solvers are used in a Predictor Corrector (PC) Method, the Predictor and the Corrector. The Predictor is a method that usually gives inferior results compared to the Corrector, also the Predictor is usually an explicit ODE solver method and is therefore computationally cheap. The Corrector on the other hand gives better results. However the price that we pay is that the methods are usually implicit and are harder to solve than explicit methods. However, by using the initial estimate from the predictor as a starting point, we can solve the implicit equations in the Corrector much more efficiently than if we had started from scratch. Therefore, the two methods work together to improve computational efficiency and accuracy.

Notice that the formulae of the two previously described ODE solver classes

(Bulirsch-Stoer, ERK) to advance a point from t_n to $t_n + h$ only involves intermediate points between the two times. Information from previous steps are not used at all. The Adams methods uses information from previously evaluated points to advance our dependent variables, thus drastically reducing the number of evaluations of the RHS of Eq.2.4. This is highly desirable because evaluations of the velocity and the acceleration can be computationally expensive and we may thus want to minimize these evaluations.

The idea of the Multi-Step Methods is as follows, given a first order ODE system,

$$\frac{dy}{dx} = f(x, y) \tag{2.62}$$

we desire to advance the dependent variable y from the point $x = x_i$ to the point $x = x_{i+1}$

In exact form

$$y(x_{i+1}) = y(x_i) + \int_{x_i}^{x_{i+1}} f(x, y(x))dx \tag{2.63}$$

. Assume that we have evaluated the value of y at k points x_{i-j}, where the sequence of x values are uniformly separated by the step size h, i.e. $x_{i-j} = x_i - jh$. We denote the numerical estimate of $y(x_{i-j})$ as y_{i-j}. WE then estimate $f(x, y(x))$ by an interpolating function

$$f(x, y(x)) \approx \sum_{j=0}^{k-1} f(x_{i-j}, y_{i-j}) L_j(x) \tag{2.64}$$

. Where k is the number of points that we use to interpolate f, y_{i-j} are numerical estimates of y at $x = x_{i-j}$ and $L_j(x)$ are the k-1 order Lagrange Interpolating Polynomials given by

$$L_j(x) = \prod_{j=0, j\neq 1}^{k-1} \frac{x - x_l}{x_j - x_l} \tag{2.65}$$

.

By inserting Eq.2.64 into Eq2.63 we obtain the explicit numerical method

$$y_{i+1} = y_i + h \sum_{j=0}^{k-1} \beta_j f(x_{i-j}, y_{i-j}) \tag{2.66}$$

. and

$$\beta_j = \frac{1}{h} \int_{x_i}^{x_{i+1}} L_j(x) dx \tag{2.67}$$

. Methods of this form are called Adams-Bashford Method and various versions of this method can be obtained by choosing different values of k.

applying Eq.2.66 to solve Eq.2.4 to advance our dependent variables **V** and **X** from the time t_n to $t_n + h$ and setting k=3, we obtain the formula for the 3-step Adams Bashford method(AB3)

V and **X** from the time t_n to $t_n + h$

$$\begin{aligned} \begin{pmatrix} \mathbf{V_{n+1}} \\ \mathbf{X_{n+1}} \end{pmatrix} &= \begin{pmatrix} \mathbf{V_n} \\ \mathbf{X_n} \end{pmatrix} + \frac{h}{12}\Bigg(23 \begin{pmatrix} \mathbf{A}(\mathbf{z_n}, t_n) \\ \mathbf{V}(t_n) \end{pmatrix} - 16 \begin{pmatrix} \mathbf{A}(\mathbf{z_{n-1}}, t_n - h) \\ \mathbf{V}(t_n - h) \end{pmatrix} \\ &+ \ 5 \begin{pmatrix} \mathbf{A}(\mathbf{z_{n-2}}, t_n - 2h) \\ \mathbf{V}(t_n - 2h) \end{pmatrix}\Bigg) \end{aligned} \tag{2.68}$$

This equation therefore will yield the "prediction value". We now turn to the Corrector Method.

The interpolation of f in Eq.2.64 did not involve the point y_{x+1}, therefore, the points used for the interpolation do not span across the entire interval of integration, to improve on this we can therefore instead use the interpolating polynomial.

$$f(x, y(x)) \approx \sum_{j=0}^{k-1} f(x_{i+1-j}, y_{i+1-j}) L_j(x) \tag{2.69}$$

. Note, this time $L_j(x)$ is a k order interpolating polynomial. The derived numerical method is therefore,

$$y_{i+1} = y_i + h \sum_{j=0}^{k} \beta_j f(x_{i+1-j}, y_{i+1-j}) \tag{2.70}$$

. and

$$\beta_j = \frac{1}{h} \int_{x_i}^{x_{i+1}} L_j(x) dx \tag{2.71}$$

.

A method of this form is known as an Adams-Moulton Method. However, the price we pay for the superior polynomial interpolation in Eq2.69 is that Eq.2.70 is now implicit. I.e. y_{i+1}, the dependent variable is on both sides of the equation. Thus these methods are harder to solve than the explicit Adams-Bashford Methods.

The corresponding 3 step Adams-Moulton, applied to Eq 2.4 is given by the implicit equation. Computation"[17]

$$\begin{pmatrix} \mathbf{V_{n+1}} \\ \mathbf{X_{n+1}} \end{pmatrix} = \begin{pmatrix} \mathbf{V_n} \\ \mathbf{X_n} \end{pmatrix} + \frac{h}{12}(5 \begin{pmatrix} \mathbf{A}(\mathbf{z_{n+1}}, t_n + h) \\ \mathbf{V}(t_n + h) \end{pmatrix} + 8 \begin{pmatrix} \mathbf{A}(\mathbf{z_n}, t_n) \\ \mathbf{V}(t_n) \end{pmatrix} - \begin{pmatrix} \mathbf{A}(\mathbf{z_{n-1}}, t_n - h) \\ \mathbf{V}(t_n - h) \end{pmatrix}) \tag{2.72}$$

Finally, we put together the Adams Bashford Method and the Adams Moulton Method in the following manner.

P.) We use Eq.2.68 to generate the estimate $\mathbf{V_n}$ and $\mathbf{X_n}$

R.) We calculate $\mathbf{A}(\mathbf{X_n}, t_n + h)$ and $\mathbf{V}(t_n + h)$

C.) We use Eq.2.72 and Step 2 to generate a refined estimate $\mathbf{V_n}$ and $\mathbf{X_n}$.

There are some variations to implement this procedure. We can either choose to end the algorithm at R or at C. In addition, we can iteratively do the steps R and C a fixed m number of times. I.e., we can have $P(RC)^m$ or $P(RC)^m C$. The most common implementation is $PRCR$[22].

Since Eq 2.68 and 2.72 give two different estimates of the same point, we can use these two evaluations to estimate the error, for example, the difference in the two evaluations estimates the error. With this error estimate, we can implement an adaptive step size and adaptive order algorithms to optimize efficiency and accuracy. A description of the implementation of these algorithms is given in ”Numerical Recipes”[22]. However, these adaptive algorithms are particularly difficult to program, fortunately, the Adams method with adaptive step size and order routine can be called by the *NDSolve* command in Mathematica by specifying the method ”Adams”. The FORTRAN package known as the Livermore Solver for Ordinary Differential Equations (LSODE) also implements the method with adaptive step size and order routines.

The results derived by this formula are usually superior to that obtained by Eq.2.68. As can be seen, this equation is implicit, i.e. $\mathbf{X_{n+1}}$ is both contained in the Left Hand Side (LHS) and RHS of Eq.2.72 and as such is difficult to solve. However, as the predictor step has already given us an initial estimate of $\mathbf{X_{n+1}}$, we use functional iteration to refine the evaluation of $\mathbf{X_{n+1}}$. I.e., we put the value of $\mathbf{X_{n+1}}$ obtained from Eq.2.68 into the RHS of Eq.2.72 to generate a refined estimate of $\mathbf{X_{n+1}}$.

Since Eq 2.68 and 2.72 give two different estimates of the same point, we can use these two evaluations to estimate the error. With this error estimate, we can implement an adaptive step size and adaptive order algorithms to optimize efficiency and accuracy. A description of the implementation of these algorithms is given in "Numerical Recipes"[22]. However, these adaptive algorithms are particularly difficult to program, fortunately, the Adams method with adaptive step size and order routine can be called by the *NDSolve* command in Mathematica by specifying the method "Adams". The FORTRAN package known as the Livermore Solver for Ordinary Differential Equations (LSODE) also implements the method with adaptive step size and order routines.

2.4 Backward Difference Formulas and Stiff Systems

There are two factors that the define step size of the solution of an ODE system. Accuracy defines how small the local error is, i.e. the smallness of the error that we add to the solution from one step in our ODE solver to the next. On the other hand, stability according to Shampine et. al. refers to "errors not growing in subsequent steps of our ODE solver."[23] For most problems, i.e. non-stiff systems, accuracy requirements set the step size, the previous three ODE solver classes all use adaptive step size algorithms based on accuracy requirements. However, for stiff systems, the stability dictates the step size.

An example of a simple stiff system is given in ”Numerical Recipes”[24]

$$u' = 998u + 1998v \tag{2.73}$$

$$v' = -999u - 1999v \tag{2.74}$$

where u and v are dependent variables and t is an independent variable. We define initial conditions

$$u(0) = 1 \tag{2.75}$$

$$v(0) = 0 \tag{2.76}$$

where u and v are dependent variables and

$$u' = 2e^{-t} - e^{-1000t} \tag{2.77}$$

$$v = e^{-t} + e^{-1000t} \tag{2.78}$$

Notice that as $t \to \infty$, $u, v \to 0$. Therefore for the solution to be stable, our numerical solution must converge to 0 for both variables.

For simplicity, we solve Eq 2.75 with the Explicit (or Forward) Euler Method,

$$\begin{pmatrix} u_{n+1} \\ v_{n+1} \end{pmatrix} = \begin{pmatrix} u_n \\ v_n \end{pmatrix} + h \begin{pmatrix} 998 & 1998 \\ -999 & -1999 \end{pmatrix} \begin{pmatrix} u_n \\ v_n \end{pmatrix} \equiv (\mathbf{1} + h\mathbf{C}) \begin{pmatrix} u_n \\ v_n \end{pmatrix} \tag{2.79}$$

This simplifies to

$$\begin{pmatrix} u_n \\ v_n \end{pmatrix} = (\mathbf{1} + h\mathbf{C})^n \tag{2.80}$$

It is apparent that from our requirement for stability, $(\mathbf{1} + h\mathbf{C})^n \to 0$. The only way for this to be possible is if the absolute value of all eigen-values of $(\mathbf{1} + h\mathbf{C})^n$ is less than 1. Or equivalently:

$$h < \frac{2}{\lambda_{max}} \tag{2.81}$$

Where λ_{max} is the eigen value of $\mathbf{C}$ with the largest absolute value. Therefore, the requirement of stability sets a limit to how large a step size h we can take which is solely dependent on the ODE system and independent of accuracy requirements.

The solution to this problem is to use an implicit ODE solver. Consider the implicit (backward) Euler Formula,

$$\begin{pmatrix} u_{n+1} \\ v_{n+1} \end{pmatrix} = \begin{pmatrix} u_n \\ v_n \end{pmatrix} + h \begin{pmatrix} 998 & 1998 \\ -999 & -1999 \end{pmatrix} \begin{pmatrix} u_{n+1} \\ v_{n+1} \end{pmatrix} \tag{2.82}$$

This simplifies to

$$\begin{pmatrix} u_n \\ v_n \end{pmatrix} = ((\mathbf{1} + h\mathbf{C})^{-1})^n \tag{2.83}$$

The eigenvalues of $(\mathbf{1}+h\mathbf{C})^{-1}$ are $1/(1+h\lambda)$ which is less than 1 for all step sizes h. Therefore, our requirement for stability is automatically met and we are free to choose our step size on accuracy requirements. The implicit euler's method is the simplest example of the Backward Differentiation Formulas (BDF).

Another appropriate Backward Difference Formula is the second order version. Applying this to solve Eq2.4, we obtain:

$$\begin{pmatrix} \mathbf{V_{n+1}} \\ \mathbf{X_{n+1}} \end{pmatrix} = \frac{4}{3}\begin{pmatrix} \mathbf{V_n} \\ \mathbf{X_n} \end{pmatrix} - \frac{1}{3}\begin{pmatrix} \mathbf{V_{n-1}} \\ \mathbf{X_{n-1}} \end{pmatrix} + \frac{2}{3}(\begin{pmatrix} \mathbf{A}(\mathbf{X_{n+1}}, t_n + h) \\ \mathbf{V}_{n+1} \end{pmatrix}) \tag{2.84}$$

The price we pay for the stability is that we now need to solve an implicit equation at every step, this is computationally expensive.

For efficiency many ODE solving programs come equipped with a stiffness detector which can automatically switch between a non-stiff solver and a stiff solver dynamically. For example, Mathematica's *NDSolve* uses the Adams Predictor Corrector Method and switches to the BDF formulas with adaptive step-size and adaptive order when the method detects stiffness.

2.5 Comparison between ODE Solver Methods

We choose the Bulirsch-Stoer method as our chief ODE solver for two reasons.

1.) Superior computational efficiency.

2.) Superior accuracy of the final numerical solution

To justify this, we compare the relative merits of the ODE solvers. In order to do so, we calculated the motion of the ion in a time varying potential that takes the ion from trapping zone d in the T-trap to trapping zone i with the three different ODE solver methods that we described previously with identical precision and accuracy goals. (This shuttling protocol will be described in greater detail subsequently.)

1.) The Bulirsch-Stoer Method with adaptive step size

2.) The Explicit Runge-Kutta (ERK) Method with adaptive step size and adaptive order.

3.) The Backward Difference Formulae (BDF) methods with adaptive step sizes and adaptive order.

The three simulations were run on Mathematica 5.2 with a Debian Linux computer. The Central Processing Unit is a Dual Core 1.8GHz AMD chip and the computer runs on 2 GB of Random Access Memory. We tabulate the computer resource usage of the three ODE solvers below.

ODE solver	Ion motion data file size	Comput. time per step[s]	Computing time
Bulirsch-Stoer	18 MB	1.4	5h 54m
ERK	638 MB	0.067	37h 11m
BDF	129 MB	0.051	5h 44m

We first note that the computational time for the Bulirsch-Stoer method and the BDF method are similar and is much less than the computational time for the ERK method. Secondly, we note that the data file that contains the ion's motion derived from the Bulirsch-Stoer method is much smaller than the corresponding data files derived from the other two methods. This is desirable because in the course of designing a shuttling protocol, we may run hundreds of simulations each one producing one of these data files. Thus be choosing the correct method, we avoid having to acquire large amounts of storage space.

Therefore, by considering the usage of computational resources, the Bulirsch-Stoer method is clearly the best method.

However, there is a much more profound reason for our choice of the Bulirsch-Stoer method. If we plot the difference between the numerical calculation derived from the BDF method and the Bulirsch-Stoer Method, we see from Fig 2.3 that there is a significant difference in the numerical estimates of the ion's motion based on the method.

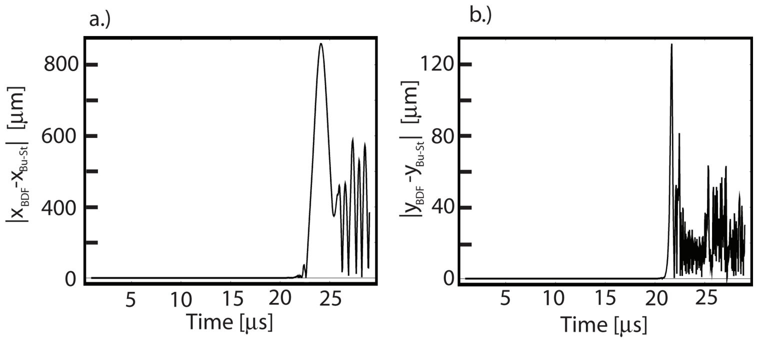

Fig. 2.3: Fig a.) Shows the absolute value of the difference between the numerical estimates of the horizontal position of the ion derived from the Bulirsch-Stoer and the horizontal position estimated by the BDF method. The magnitude of this residual is comparable to the extent of the horizontal motion of the ion $1000\mu m$. Fig b.) depicts the absolute value of the difference between the numerical estimates of the vertical position of the ion derived from the Bulirsch-Stoer and the vertical position estimated by the BDF method. Again the residual is comparable to the extent of the ion's vertical motion $500\mu m$. We note that the residual dramatically increases at $t = 20\mu s$. This corresponds to the moment when the ion enters the junction region. Before this time, the residual is very nearly 0.

To see why this is true, we first consider Fig 2.4 which represents the potential that the ion sees when it first enters the junction. At this time there are two minima corresponding to positions f and i. As such, once the ion is sent into the junction region, it will move towards either trapping region f or i. The potential gradients in the junction region are very small and therefore the ion's choice with respect to which minima it moves towards depends very strongly on the velocity of the ion once it reaches the junction region. This process thus magnifies any small error in the position and velocity of the ion just before the ion enters the junction region. Therefore, it is of critical importance to reduce the error of the motion of the ion.

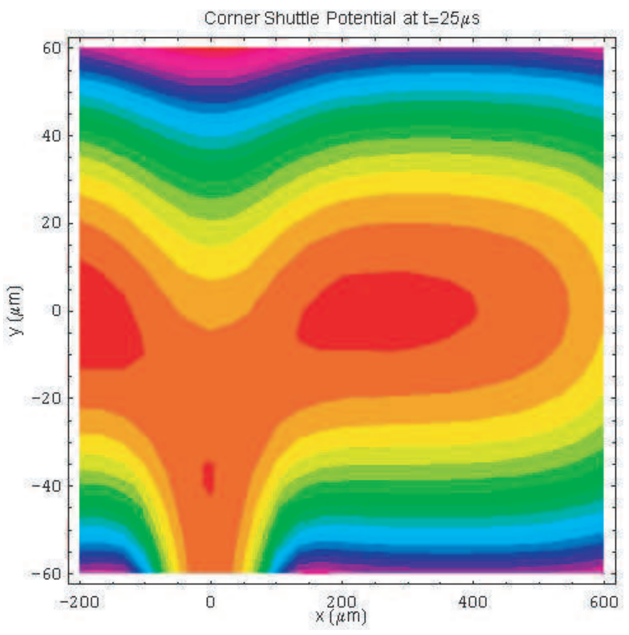

Fig. 2.4: Contour plot of the potential near the T-junction at t = 25μs. Along the x-axis there is a local potential maximum. This implies that once the ion enters the junction region, the ion will choose to move either towards the minimum on the left or the minimum towards the right. Furthermore, the ion's 'decision' is sensitive to initial conditions.

It turns out that the numerical ODE solver that takes the least number of steps provides the most accurate final solution.

To see why this is so, we consider the inner working of these numerical ODE solver methods. At every step of an ODE solver, we calculate the node y_{n+1} from the previous node y_n. The solver estimates the error of y_{n+1} as described in previous sections. This error estimate is the *local* error estimate and it is defined to be the error of y_{n+1} if y_n were the exact solution. Thus, the random error that is introduced at every step is independent of the error from the previous step. This is similar to a random walk and therefore, on average, the error increases monotonically with the number of steps taken. The adaptive step size algorithm changes the step size such that the local error estimate for each step is smaller than the error goals that we specify. Therefore, we expect that given a set of error goals, the average error introduced per step is the same regardless of the ODE solver method. This implies that given two numerical ODE solvers that numerically solve an ODE system, the ODE solver that takes less steps will usually be more accurate than the ODE solver that takes more steps.

As an example, we compare the BDF method and the Bulirsch-Stoer Method in solving the simple ODE given by.

$$\begin{aligned} \frac{d^2y}{dt^2} &= -\omega^2 y \qquad (2.85) \\ y(0) &= 0 \end{aligned}$$

. The exact solution is given by

$$y = \cos(\omega t) \tag{2.86}$$

. We set $\omega = 10^6$. We then use the Bulirsch-Stoer Method and the Backward Difference Formulae to numerically evaluate the solution for the time interval $(t = 0, t = 0.01)$. We first observe that the BDF method takes more steps than the Bulirsch-Stoer Method; 958331 steps as compared to 207422.

The second observation is that the average deviation from the exact solution increases monotonically and approximately linearly with time. The linear behavior of the error is an artifact of the simple form of the ODE. The monotonically increasing error is a feature that is to be expected regardless of the ODE system to be solved.

And lastly, if we ignore the spurious errors due to the interpolation process, the error of the Bulirsch-Stoer method is much smaller than that of the BDF method and verifies our claim that an ODE solver that can cross the interval in less steps will be more accurate than an ODE solver that crosses the interval in more steps.

We now return to the numerical solution of Equation 2.4. We tabulate data characterizing the step sizes of the ODE solver.

From table 2.5 , we see that on average the BDF method takes more than 10 steps to cross the time interval covered by a single Bulirsch-Stoer step, whilst the ERK method takes on average 60 steps to cover the time interval that a single Bulirsch-Stoer step takes. Therefore, we expect that the Burlirsch-Stoer

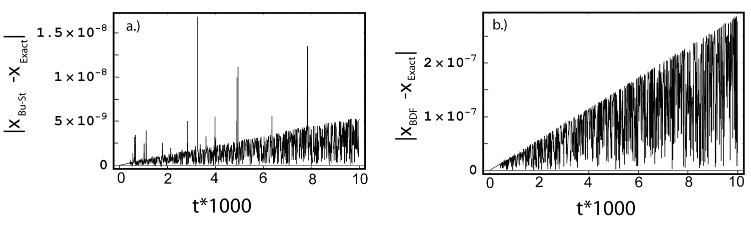

Fig. 2.5: Fig a.) depicts the absolute value of the deviation of the numerical estimate derived from the Bulirsch-Stoer method and the exact solution. There are several spurious peaks in the graph and these are due to inaccuracies in the interpolation process to fit the generated nodes and are not errors from the numerical solution. If we ignore these spurious peaks, we note that the average error increases linearly with time. Fig b.) depicts the absolute value of the deviation of the numerical estimate derived from the Bulirsch-Stoer method and the exact solution. Unlike Fig a, there are no spurious peaks because the BDF method generates more nodes which implies that the nodes are closer together and thus the interpolation process is more accurate. We note that in both figures the average error increases monotonically with time and the error of the solution derived from the Bulirsch-Stoer method is much smaller than the error derived from the BDF method.

method gives the most accurate numerical calculation of the ion's motion.

2.6 Requirements on Electric Potential

Due to the imperfect nature of the numerical evaluation of the potential, the resultant potential may be rough. This will result in relatively large arbitrarily directed forces near the potential minimum. For an ion at rest, this creates two related problems.

1.) Using the adaptive step size methods that we discussed earlier, we may need a very small step size to meet the error goals. This is computationally expensive. As an example, using the Bulirsch-Stoer method to simulate a shuttling protocol from d to i, the Bulirsch-Stoer step size at the start of the protocol is

ODE solv. method	Max (Min) step size[μs]	Number of steps	Ave step size[μs]
Bulirsch-Stoer	$1.5 * 10^{-8}$ $(7.9 * 10^{-16})$	35392	$8.8 * 10^{-10}$
ERK	$4.1 * 10^{-11}$ $(7.9 * 10^{-16})$	2013103	$1.9 * 10^{-11}$
BDF	5 $(7.9 * 10^{-16})$	408403	$7.6 * 10^-11$

Tab. 2.2: Step size and other related quantities used by various Numerical ODE solvers to solve a simple problem.

of the order of $10^{-15}s$ whereas the average Bulirsch-Stoer step size for the entire motion was $10^{-9}s$.

2.) These artificial forces may create a stiff ODE system. Stiff ODEs are both computationally expensive to evaluate and sacrifice accuracy for stability (since stiff systems are intrinsically unstable.) Therefore, the computation will be *both* slow and inaccurate.

These two problems can make solving the ODE intractable.

There are two solutions to these problems.

1.) We can swamp the artificial forces with the actual secular force. This can be achieved by slightly displacing the ion from its initial position. Equivalently, we can start the ion with a small initial velocity.

2.) We can re-derive the potential with a finer mesh.

Fig 2.4 represents the potential that the ion sees when it first enters the junction. At this time there are two minima corresponding to positions f and i. As such, once the ion is sent into the junction region, it will move towards either f or i. Simulating the ion's motion for various initial position offsets, we observe that the ion's choice with respect to which minima it moves towards depends on the initial position offset. In addition, we did not observe any simple correlation between the initial position offset and the direction with which the ion moves towards. Therefore, the initial position offset presents an unacceptable error when high accuracy solutions are desired. In particular, such an error will make it impossible to optimize the kinetic energy or design phase dependent cooling schemes.

Therefore, the only solution to this problem is to ensure that the potentials are sufficiently smooth. Therefore by requiring that the calculation of the ion's dynamics from a standing start is tractable, we can obtain a useful lower limit for the accuracy with which we solve for the potential in the trap.

3. MANIPULATION OF ATOMIC IONS IN 1 DIMENSION

With the general strategies and techniques that were developed in previous sections to develop key shuttling protocols for the T-junction Ion Trap architecture. The ion trap array has 49 electrodes which are used to implement 4 key protocols necessary to implement the quantum computing scheme proposed by Kielpinski et al[10].

1.) Shuttling an ion along an rf node between two adjacent traps.

2.) Separating two ions held in the same trap into two different traps.

3.) Combining two ions held in two separate traps into a single trap.

4.) Shuttling ions around corners.

The first three protocols have been widely demonstrated ([25],[26],[27],[28]) whilst corner shuttling was first demonstrated in 2005 on the T-Trap by Hensinger et al. [29]. The combination of all four elementary protocols allows for arbitrary control of trapped ions in two dimensions.

In this chapter, we will discuss the motion of the ion along an rf node, i.e. the first three protocols and in the next chapter we will go out into 2 dimensions and there we will describe the most difficult of the 4 key protocols, corner shuttling.

3.1 Linear Shuttling

Linear shuttling in ion traps involves the transport of ions from one trapping zone to another adjacent trapping zone along a linear rf-node. This was demonstrated by Rowe et al. [25] We begin the discussion of this type of shuttling sequence by deriving a figure of merit from the adiabatic condition.

3.1.1 Linear Shuttling and Adiabaticity

It is highly desirable for a shuttling protocol to be adiabatic. An adiabatic process is a reversible process and therefore repeated applications of the shuttling protocol will not significantly change the state of the ion. This is highly desirable in quantum computation as a quantum algorithm may require an ion to be shuttled back and forth numerous times whilst preserving its motional state throughout, therefore, we derive a criterion for the adiabaticity of shuttling an ion along a linear rf node.

We define a shuttling protocol to be adiabatic if not enough energy is added to the system to allow for transitions to other quantum states [30]. We define the wave-function of an ion to be $\Psi(t)$ and $\Psi(t=0)=\Psi_s$. We assume there exist several quantum states which are accessible to the ion Ψ_e, then for the shuttling protocol to be adiabatic, the following criterion must be satisfied.

$$\begin{aligned} |\langle\Psi(T)|\Psi_s\rangle|^2 &\approx 1 \\ |\langle\Psi(T)|\Psi_e\rangle|^2 &\approx 0 \end{aligned} \tag{3.1}$$

For this analysis, we assume that the secular frequency of trap does not change throughout the shuttling sequence.

The Hamiltonian for this system will thus be

$$H = \frac{1}{2}m\omega_0^2(x - x_0(t))^2 \tag{3.2}$$

Where x is the position of the ion and x_0 is the position of the potential minimum.

In order to transform Equation 3.1 into a working criterion, we apply first order perturbation theory.

It turns that it is mathematically convenient to get out of the rest frame of the ion trap and into the moving frame of the minimum of the rf trap, $x_0(t)$. Our Hamiltonian will now be

$$H = \frac{1}{2}m\omega_0^2(s)^2 + m\ddot{x}s \tag{3.3}$$

Where we define the new variable $s = x - x_0(t)$. The price that we pay is that there is now a virtual potential due to the acceleration of the potential minimum. We therefore immediately see that the adiabaticity of our shuttling protocol depends on the acceleration of the potential minimum and it turns out [31] that Equation3.1 implies

$$\frac{\dot{a}}{a} << \omega \tag{3.4}$$

Where a is the acceleration of the potential minimum and ω is the secular frequency. The classical analogy is that if we hold a pendulum in a car, the pendulum will swing steadily and thus adiabatically if the car is constantly accelerating. However, once we slam on the brakes or the accelerator, i.e. high $\dot{a}$, the pendulum will jerk and thus non adiabatically.

3.1.2 Implementation of Linear Shuttling protocol

We next consider an example of a linear shuttling protocol. This shuttling sequence brings an ion from trapping zone c to b. At the start of the shuttling protocol, electrodes 0, 1, 2, 3, 8, 9, 16 and 17 are all held at 25V. For convenience, we refer to these 8 electrodes as the *outer electrodes* of the shuttling sequence. Electrodes 4 and 5 also start at 25V whilst electrodes 6 and 7 start at -2.5V. The secular frequency in the x direction of the initial is calculated to be 1.2MHz. For the next 15 clock cycles, the voltages on the 8 outer electrodes monotonically increase whilst electrodes 4, 5, 6 and 7 decrease. At the midpoint between the 15th and 16th clock cycle, the values of 4,5,6 and 7 all coincide to be -18V whilst the perimeter electrodes all reach 184.5V. This implies a trap at the midpoint in between trapping zones b and c. The ramping of the voltages was so chosen such that at all points of time, the secular frequency in the x-direction remains constant. This completes the first half of the shuttling protocol.

To complete the protocol, i.e. to bring the ion from the midpoint of points b and c to b, the voltage protocol is both time reversed and geometrically reflected about the plane that would fit in the gap between electrodes 4, 5 and electrodes 6, 7. Time reversing the first half of the shuttling sequence sends the ion away from the midpoint into one of the trapping zones whilst the geometric reflection ensures that the ion moves from the midpoint to b instead of from the midpoint to c. See Figure 3.1. The symmetry of the second half of the shuttling protocol also implies that the secular frequency is constant throughout the second half of the protocol and the same as the first half of the shuttling protocol.

This linear shuttling protocol with a fixed secular frequency is the first step to optimizing our linear shuttling protocol. We test the above described shuttling protocol on the T-Trap, to do so we use the shuttling protocol to shuttle the ion from c to b, then run the protocol backwards in time to shuttle the ion from b to c. The ion successfully completes the protocol every time (55 attempts) over a wide range of shuttling speeds. The time elapsed for the shuttling protocol ranges from 6s to 6×10^{-5}s. This corresponds to a clock rate of 10Hz to 10,000Hz.

Furthermore, with this shuttling protocol we could reliably run the shuttling protocol back and forth ten times in succession with a clock rate of 10,000 Hz. (10 out of 10 attempts.) Each shuttling sequence iteration begins with the clock cycle immediately after the last clock cycle of the previous shuttling iteration.

In contrast, crude linear shuttling protocols that were used to demonstrate arbitrary two dimensional control of ions [29] could not linear shuttle back and forth multiple times. Therefore, constraining the secular frequency dramatically improves the performance of linear shuttling protocols.

Presently, work is being done to further refine this linear shuttling protocol by using the criterion given by Equation 3.4.

3.2 Ion Separation and Recombination

The next two key shuttling protocol that we discuss are separation and recombination. The goal of separation is to separate two or more ions held in a single harmonic trap into two new harmonic traps. The goal of recombination is the

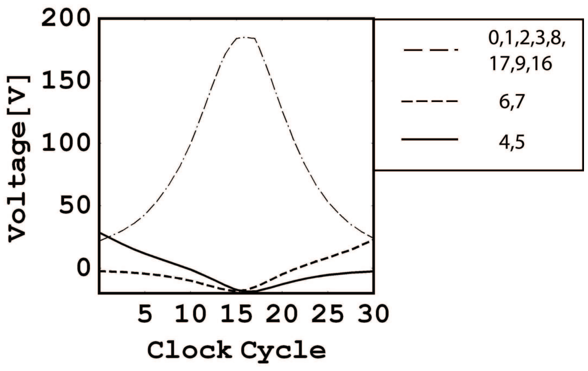

Fig. 3.1: The graph shows the time dependent voltages applied on the relevant electrodes which shuttle the ion along the linear node from trapping zone *c* to *b*. The secular frequency in the x direction is constrained to be 1.2 MHz at all times in the shuttling sequence.

exact opposite, we start with two harmonic traps each containing some ions and combine all the ions into one new harmonic trap. Note that in separation and recombination, we do not change the order of the ions.

We consider a specific example of a separation protocol. We begin by trapping two ions in a harmonic trap in zone *b* with approximate secular frequency 0.53MHz. The trap is formed by applying -3.5V on electrodes 4 and 5, and 17 volts on electrodes 2, 3, 6 and 7. In addition, 17 volts are applied to electrodes 0, 1, 8 and 17.

For the next 30 clock cycles, the voltages on electrodes 2, 3, 6 and 7 gradually ramp down to -3.5V whilst the other electrodes do not significantly change, this in effect changes weakens the trap at *b* such that the secular frequency is now 0.099kHz. More importantly, the potential at trapping zone *b* is very nearly flat. Because of this, the Coulomb repulsion between the two like charged ions is now significant and we calculate a separation between the two ions of approximately $10\mu m$.

From here, we ramp up the voltage on electrodes 4 and 5 to 17.5V. This creates a potential wedge at zone b and sends one ion into zone a and the other ion into zone c. Electrodes 0,1, and 8 and 17 are maintained at 17V. This provides the necessary confinement to form harmonic traps at zones a and c.

Next, we consider an example of a recombination protocol. The recombination protocol is the separation protocol time reversed. We start with an ion held in a harmonic trap in trapping zone a and another ion in trapping zone c. The harmonic traps at a is formed by high voltages on control electrodes 0,1,4,5,8 and 17, and low voltages on electrodes 2,3,6 and 7. The voltage on electrodes 4 and 5 are brought down to the same level as electrodes 2,3,6 and 7. This forms a weak trap at zone b. At this point, the ions will be close together, $10\mu m$ apart. To complete the combination protocol, we strengthen the trap at b by raising electrodes 2,3, 6 and 7. Thus, we now have two ions in a harmonic trap at zone b.

3.2.1 Experimental Implementation

The separation and recombination protocol was implemented with a shuttling time of 10ms. However, the success rate was only 58% over 64 attempts. There are two possible reasons for this low efficiency.[25]

1.) During the separation sequence, we require the wedge to come up in between the two ions. This is difficult due to the small separation between the ions to be separated and the large width of the control electrodes. In this case,

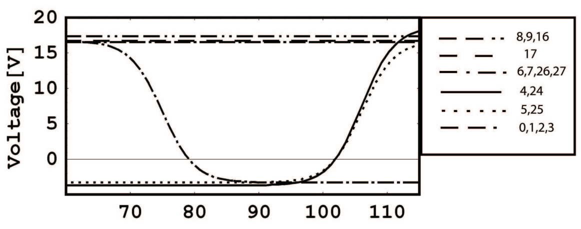

Fig. 3.2: Time dependent voltages applied on control electrodes in order to separate two ions held in a single harmonic trap at zone b into two separate harmonic traps, one at a and the other at c. Asymmetries in the voltage profile about the x=0 plane are to compensate for possible misalignments in the electrode layers or stray fields. A time reversal of the above voltage profile allows us to combine two ions, one in a trapping zone a and the other at c into a single harmonic trap at zone b.

both ions will be shuttled into the same trap.

2.) During the separation sequence, there will be a time where the ions will be held in a weak trap at trapping zone b. This occurs just before the wedge is brought up. During this time, the ions are especially susceptible to heating.

As we can see, the two types of failures provide competing requirements on the separation protocol. The first failure can be addressed by increasing the ion spacing, however, increasing the ion spacing is achieved by further weakening the trap at zone b, thus further heating the ion. Therefore, balancing these two demands is essential to the success of a separation protocol.

4. MANIPULATION OF ATOMIC IONS IN 2 DIMENSIONS

In section 1.2.1, the linear nodes in each of the three linear arms gives way to humps, i.e. regions of no rf field cancelation, as we approach the junction region. For all the protocols mentioned in the previous chapters, the ion never leaves the rf node. This is no longer true once the ion goes through the junction region as the ion must now be pushed over the rf humps. Therefore, the design of shuttling protocols described in this chapter is a much more involved process and is the subject of this chapter.

4.1 Shuttling around the Corner

In a general junction shuttling protocol, there are two stages.

1.) We first overcome the inevitable RF humps leading into the junction by providing a sufficient pushing potential and shuttle the ion into the junction.

2.) We guide the ion from the junction region into its final trapping position

The Tee-Trap corner shuttling protocol that brings the ion from trapping zone d to i is an exemplary example of such a protocol. On calculating the ponderomotive potential in the Tee-trap, the three linear RF nodes which provide strong axial confinement give way to RF humps of height 0.1 eV as we approach the junction..

The 4 electrodes that form the junction, i.e. 8, 9, 16 and 17 are the most important and thus the success of the shuttling protocol greatly depends on the time dependent voltage on these 4 electrodes, we shall probe the stability of the voltage time profiles on these 4 electrodes in the subsequent section. In addition, adjacent electrodes to the junction; 6, 7, 26, 27, 10, 11, also play a part in the shuttling protocol.

We begin with a trap in region *d*. The trap frequencies at this point are

$$\begin{aligned} \omega_x &\approx 5.0MHz \\ \omega_y &\approx 0.7MHz \\ \omega_z &\approx 4.9MHz \end{aligned} \tag{4.1}$$

and the trap depth is 12 eV. Using a hyperbolic tangent profile, we raise 6, 7, 26 and 27 to 200V. This effectively pushes the ion towards the junction region. The value of 200V is determined solely by the electrical properties of the trap. We simultaneously raise electrodes 8 and 17 from approximately -4V to ground and lower 9 and 16 from approximately $80V$ to $-3V$. We again use a hyperbolic tangent time profile. The net effect is to create a potential gradient that sucks the ion into the junction region. We can observe the position of the RF hump through the y component of the ion's trajectory. The ion temporarily gets stuck at some barrier at $y = -220\mu m$ at the 18th clock cycle. This shows that the ion is up against the RF hump. We also notice that the micro-motion abruptly increases. Up till this point, the kinetic energy of the ion monotonically increases from 0 to approximately 0.07 eV.

At the 19th clock cycle, the potential gradient becomes sufficient such that the ion can roll over the hump. The ion accelerates from $y = -220\mu m$ at the 19th clock cycle to $y = 0$ in the period of $2\mu s$. In the process of rolling down the potential hill, the ion picks up approximately 0.5 eV of energy. The ion then bounces off the RF layer and moves towards the negative y direction at $y = 0$. Once there the ion gains freedom of motion in the x direction and it will oscillate wildly between regions *f*, e, i with the energy of approximately 0.5 eV. This large oscillation is mainly due to there being very weak confinement along the x-axis. As shown in section 2.6, the motion in the x direction strongly depends on the initial conditions. Therefore,the ion's starting position in subsequent numerical simulation to determine the protocol to execute step 2 should not include any arbitrary offsets from the initial equilibrium. In addition, the ion experiences micro-motion as it transverses the hump. Therefore, high accuracy solutions are now needed.

Notice that the voltages on 8, 17, 9 and 16 all converge to voltages close to ground. This is because high static voltages on the four junction electrodes can make e anti-trapping and thus cause the protocol to fail. Therefore, there is a tradeoff when designing such a junction shuttling protocol. We require a high potential gradient and thus a high absolute value of voltages to overcome the RF hump send the ion into the junction region. At the same time, the voltages must be low enough such that the junction region can remain trapping. Therefore, the key to the success of this step is to balance these two requirements.

To implement step 2 in the Tee-Trap, we now need to trap the ion at i. In order to do so, we raise the voltages of 8 and 9 to approximately 10 V and lower the voltages of 16 and 17 to approximately -10V. Prior to this, 10 and 11 were raised to 100V. This thus forms the other side of the trap. The trap has frequencies

$$\begin{aligned} \omega_x/2\pi &\approx 0.5MHz \qquad (4.2) \\ \omega_y/2\pi &\approx 5.5MHz \\ \omega_z/2\pi &\approx 4.3MHz \end{aligned}$$

However, instead of implementing the change in voltage by the usual hyperbolic tangent, we implement the change in voltage of the four junction electrodes as quickly as possible, i.e. one clock cycle of our analogue output card. We denote this clock cycle to be t_{catch}. We choose to catch the ion in such a fashion for the following reasons.

1.) We optimize the shuttling sequence such the ion is 'caught' by the final trap when it is exactly at its final equilibrium position. This would add no further kinetic energy to the ion. Using any other kind of time profile to form the trap will yield inferior time definition of the catching step.

2.) As t_{catch} is a very simple variable to perturb, a large number of variations of a shuttling protocol can be easily produced by changing t_{catch}. This is useful because changing t_{catch} addresses the uncertainty in the x position between steps 1 and 2, demonstrated in section arbitrary offsets.

For this particular shuttling protocol, the act of catching the ion in the final trap raises the kinetic energy by approximately 0.7eV as can be seen by the abrupt jump in kinetic energy at the 25th clock cycle coinciding with t_{catch}.

We note that numerical simulations show that implementing the shuttling protocol with a hyperbolic tangent profile may also work, however, due to the aforementioned reasons, optimizing this protocol is difficult and thus this style of protocol was not well studied. Finally, following from the discussions of ODE solving methods, we suggest that in designing step 1, the ERK method be used. This is because low accuracy solutions will suffice to calculate the ion's path from position d to i. However, in designing step 2, since much more accurate solutions are required, particularly to model the ion's motion in the junction, we recommend using the Bulirsch-Stoer method.

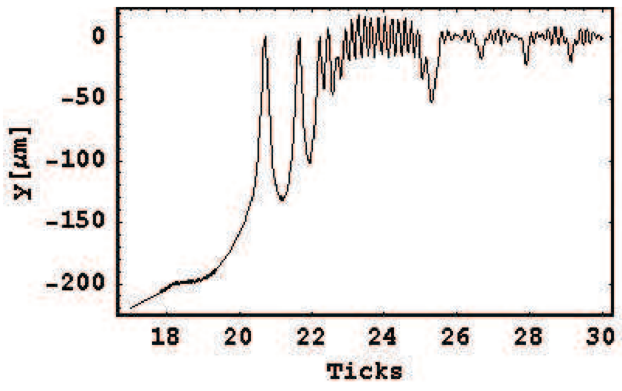

Fig. 4.1: Figure shows y component of motion when ion is near and inside the junction region, note the micro-motion at $t = 18\mu s$ to $t = 20\mu s$ as the ion transverses the RF hump.

4.1.1 Experimental Results

The control voltage protocol was used to shuttle the ion from region d to region i in the T-Trap and was implemented with a success rate of greater than 99%. [29](881 out of 882 attempts). The speed of the shuttling protocol was limited

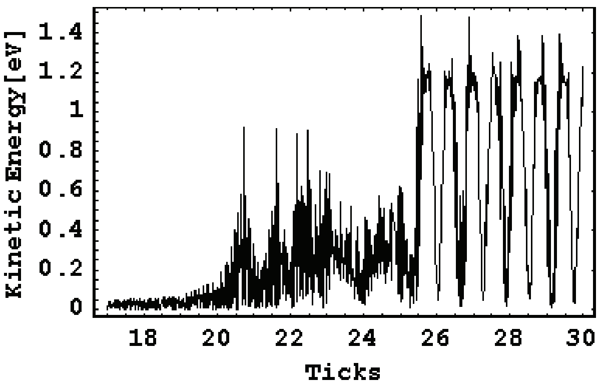

Fig. 4.2: Figure shows kinetic energy. Note the rise in kinetic energy once the ion crosses the RF hump at the 18th clock cycle and the abrupt rise in energy at the 25th clock cycle when the t_{catch} step is implemented.

by the hardware and as such the fastest shuttling protocol was run at a clock rate of 10^6 cycles per second. This corresponds to a shuttling time of $30\mu s$.

We implemented the shuttling protocol with different clock rates and successfully turned the corner with a range of clock rates from 0.1 cycles per second to 10^6 cycles per second. The upper limit to the clock rate, as previously mentioned, is due to hardware limitations. The clock rate of 0.1 cycles per second is not a lower limit as we did not test slower clock rates due to time constraints. This is because the ion takes five minutes to crystallize and bringing down the shuttling times by an order of magnitude would imply a shuttling time of 1 hour.

Despite the very high success rate of the shuttling protocol from d to i, we were unsuccessful in corner shuttling the ion from d to f. This result is surprising since turning the corner from d to i should immediately imply that turning the corner from d to f should work. This discrepancy may be attributed to static bias fields or misalignments in the three electrode layers due to the manual

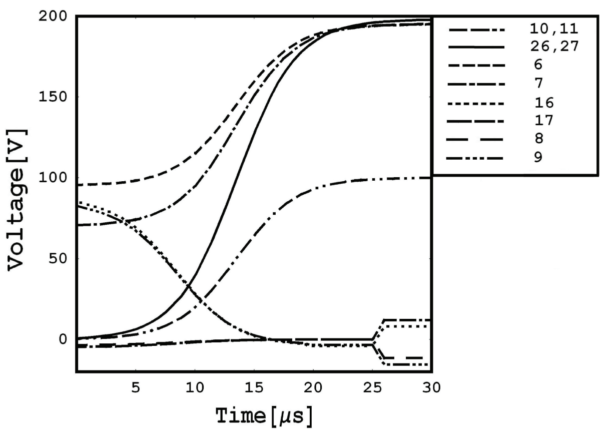

Fig. 4.3: Figure shows the voltages applied to the relevant electrodes to execute the corner shuttling protocol.

assembly process.

4.1.2 Characterization of Corner Shuttling Protocols

To characterize our corner shuttling protocol, we perturb the control voltage protocol and simulate the ion's motion. We will achieve two things by doing so.

1.) Many of these perturbed voltage protocols will not successfully shuttle the ion from d to i. From these failed protocols, we can infer salient features that lead to a successful shuttling protocol.

2.) We can probe the robustness of the shuttling protocol to small perturbations.

The four junction electrodes of the Tee-trap, 8, 9,16 and 17 are the most important electrodes in the shuttling protocol. Thus, our perturbations are generated in the following manner. We begin by choosing one of the 4 junction electrodes to alter, for example electrode 17. The time varying voltage on that electrode is given by the function $\Phi_{17}(t)$. Here t denotes the clock cycle. We leave the voltage profile after the t_{catch} clock unaltered. For the preceding points, we define them by

$$V_{17,V_j}(t) = \begin{cases} A\tanh(\frac{t-7.5}{5}) + B & 1 \leq t \leq 20 \\ V_j & 21 \leq t \leq 25 \\ V_{17}(t) & 26 \leq t \end{cases} \tag{4.3}$$

Where A and B are constants such that $V_{17,V_j}(0) = V_{17,V_j}(0)$ and $V_{17,V_j}(20) = V_j$. With this definition, the voltage profile smoothly asymptotes to V_j and also ensures that at the first clock cycle, the potential on the electrode is the same for the base protocol and the altered protocol.

We generate perturbations for each of the 4 junction electrodes for several values of V_j and numerically simulate the path of the ion. We use the inverse of the kinetic energy at the end of the protocol as our figure of merit and define this Figure of Merit to be zero if the protocol fails. For comparison, the Figure of Merit for the unaltered protocol is shown in each graph. Therefore, for the following plots, a higher inverse energy implies a better corner shuttling protocol.

We numerically simulate these generated shuttling protocols and obtain the following stability plots.

From the numerical simulations, we see that there are three ways that a shuttling protocol may fail.

1.) The ion does not make it over the RF hump and is stuck in the original trapping region.

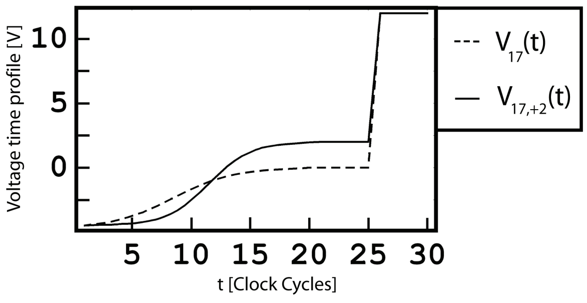

Fig. 4.4: Time profile of voltage on electrode 17 in original shuttling protocol that brings ion from d to i and ion plotted with the perturbation $tV_{17,+2}$

2.) The ion is shuttled into the junction region but is shortly ejected from the plane of the trapping region

3.) The ion goes too far in the negative x direction (region g or beyond) and is ejected out of the trap in the negative x direction at the t_{catch} step.

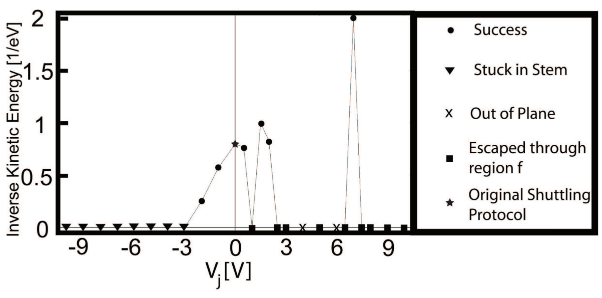

Fig. 4.5: Stability of shuttling protocol to changes in voltage applied to Electrode 8.

Repeating the process for the other three electrodes we obtain

We observe that for all stability plots, there is a cut off voltage that separates all protocols that shuttle the ion into the junction region and all protocols that leave the ion at its original trap position (failure type II).

For electrodes 8 and 17, we notice that the ion is stuck for perturbations

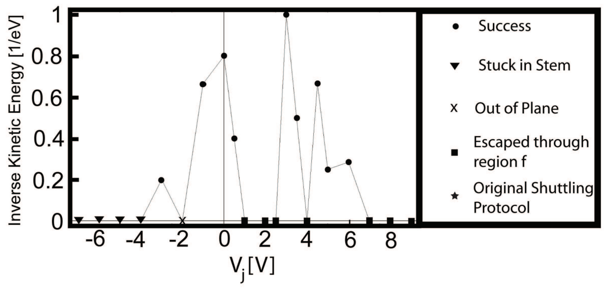

Fig. 4.6: Stability of shuttling protocol to changes in voltage applied to Electrode 17.

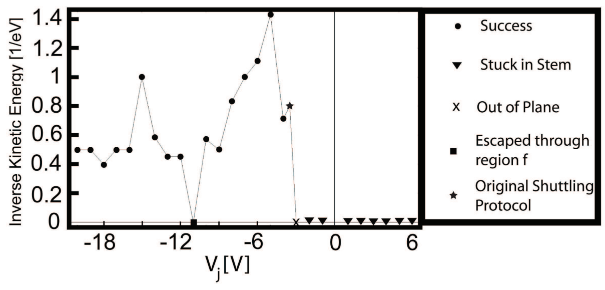

Fig. 4.7: Stability of shuttling protocol to changes in voltage applied to Electrode 9.

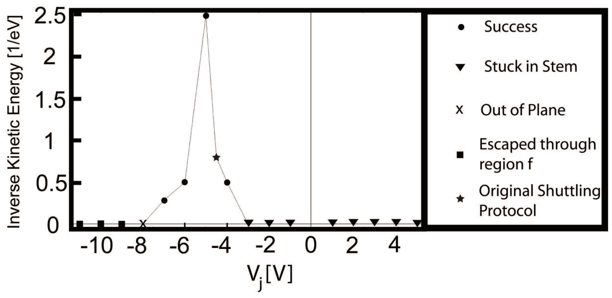

Fig. 4.8: Stability of shuttling protocol to changes in voltage applied to Electrode 16.

that lower the electrode voltage. This is consistent with what we expect as a negative voltage on these lower electrodes will 'suck' the ion back into its original trap. Similarly, for the two upper electrodes, 9 and 16, this occurs for perturbations that raise the electrode voltage. In effect, the upper electrodes repel the ion away from the junction and back into the trap

Therefore, when the ion is stuck in the original trap region, it would indicate that there is insufficient potential gradient for the ion to overcome the RF hump.

Next, we notice that the Stability plot for electrode 9 has a much greater stability range for negative voltage perturbations than for the other three electrodes. There are several reasons for why this is so. For one, the large negative voltage sucks the ion into the junction region. Thus, the ion does not get stuck in the original trapping position. Secondly, the ion is strongly attracted towards its final trapping position at i, therefore, the ion will spend no time in region d, therefore, there is no way that the ion will be ejected from the trap from the negative x direction. Lastly, as the ion can only be in region i at t_{catch}, the t_{catch} trapping step will add very little kinetic energy to the ion. Therefore, all these factors add to the stability of the shuttling protocol. However, shuttling protocols of this nature have certain disadvantages, firstly, once the ion is inside the junction region, it will inevitably gain a set amount of energy as it rolls down the potential hill from region e to i. From figure (DC9 stability), we observe that the final kinetic energy plateaus at 2.5 eV. This is a factor of 2 greater than that from the default protocol. In addition, due to the large imbalance in the potential of electrodes 8 and 9, the ion will not sit at the RF node at i and will therefore experience severe micro-motion.

The third observation is protocols obtained by perturbing electrodes 9 and 16, each has one region of consistent stability. On the other hand, the protocols obtained via perturbations on electrode 8 and 17 have no such clear region of stability. The reason for this is that once the ion has crossed the RF hump, the axial confinement in the x direction abruptly decreases. This occurs at $y = -20\mu m$. At this point, the ion is much nearer to electrodes 8 and 17 as compared to electrodes 9 and 16. This means that the motion of the ion inside the junction strongly depends on the voltage of these two electrodes in an unstable manner.

4.1.3 Adiabaticity of Corner Shuttling

It is apparent from section 4.1 that turning the corner in the T-trap is non-adiabatic. The non-adiabaticity stems from the moment when the ion just overcomes the rf hump. Once the control electrodes provide sufficient pushing potential gradient, the ion abruptly rolls down a potential hill into another minimum in the middle of the junction. However, in order for a shuttling protocol to be adiabatic, the ion needs to be near the minimum of a strong harmonic potential at all times. Therefore, the corner turning protocol is non adiabatic.

One potential solution to this problem is to use the junction electrodes to swamp the effect of the rf hump. This can be achieved by using large enough static voltages. I.e. electrodes 9, 16, 4, and 5 need to be held high compared to electrode 8 and 17 throughout the shuttling process from region d to the junction region e.

However, the stability analysis in section 4.1.2 shows that the shuttling protocol will fail if the voltages on 9 and 16 are high when the ion is inside the junction region. It is therefore likely that new ion trap junction geometries which reduce or eliminate the rf hump will be needed to implement adiabatic corner turning.

4.2 Shuttling around the Corner the other way

The shuttling sequence from trapping zone i to trapping zone d is similar to the shuttling protocol from zones d to i. As such, there are two steps to execute such a shuttling protocol.

1.) We need to use the control electrodes to overcome the rf hump and shuttle the ion into the junction.

2.) From the junction region, we guide the ion into its final trapping position.

This shuttling protocol is obtained by spatially reflecting the shuttling protocol that brings the ion from zone d to zone i about the line joining electrodes 8 and 16. Thus we swapped the time dependent voltages on the critical electrodes in the following manner.

$$\begin{aligned} 9 &\leftrightarrow 17 \\ 10 &\leftrightarrow 6 \\ 11 &\leftrightarrow 7 \\ 8 &\leftrightarrow 8 \\ 16 &\leftrightarrow 16 \end{aligned} \tag{4.4}$$

Electrodes 8 and 16 keep the same time dependent voltage profile.

We thus begin with a trap at region i with secular frequencies

$$\omega_x \approx 0.7MHz \tag{4.5}$$

$$\omega_y \approx 4.5MHz \tag{4.6}$$

$$\omega_z \approx 4.2MHz \tag{4.7}$$

Electrodes 10 and 11 ramp up via a hyperbolic tangent profile to 200V. This provides a pushing force into the junction region. Simultaneously, we raise electrodes 8 and 9 from $-4V$ to ground and lower electrodes 16 and 17 from $35V$ to $-4V$. The configuration of these four electrodes provides an effective potential gradient that sucks the ion from its starting point between electrodes 8 and 9 into the junction region. Again, as the voltages gradually change, the ion slowly moves closer to the junction and at the 15th clock cycle, gets stuck behind the RF hump for 1 clock cycle. This can be seen in Figure 4.10 as the ion gets stuck behind the rf hump at $x = 200\mu m$. We note that the micro-motion of the ion severely increases during this clock cycle, further indicating that the ion is pressed up against the rf hump. Up till this point, the ion's kinetic energy increases monotonically from 0 to approximately 0.02 eV.

At the 16th clock cycle, the potential gradient abruptly becomes sufficient to overcome the rf hump and the ion rolls down a potential hill into the junction region. The ion takes approximately one micro-second to roll down the hill and gains approximately 1eV of kinetic energy in the process.

Once inside the junction region, the ion is free to make large scale oscillations due to the small electric fields in the junction region and the kinetic energy it acquired in overcoming the rf hump. During this time, the ion can access regions d, e and f.

To complete the shuttling sequence, we "catch" the ion by ramping the voltages of the control electrodes into the final trapping configuration in a single clock cycle, again we denote this clock cycle as t_{catch}. Electrodes 8 and 17 drop to approximately -15V and electrodes 9 and 16 ramp up to approximately 10V. This results in the final trap at region d with secular frequencies

$$\omega_x \approx 5.0MHz \tag{4.8}$$

$$\omega_y \approx 0.6MHz \tag{4.9}$$

$$\omega_z \approx 3.6MHz \tag{4.10}$$

Ideally, the t_{catch} step should occur when the ion is in region d, in this way, the ion gains no further kinetic energy from step 2. Thus in the design of our shuttling protocol, we allowed easy adjustment of the value of t_{catch}.

4.2.1 Experimental results

Using the above mentioned control voltage protocol, we shuttled the ion around the corner from zones i to d. The success of this shuttling protocol was only 98% (118 attempts). Unlike the forward shuttling protocol, this corner turning protocol could only be run at a maximum of 1800 clock cycles per second. Recall that the fastest shuttling time from d to i is limited by the hardware. The fastest time from i to d was therefore to the order of 20 ms. At any higher output rate,

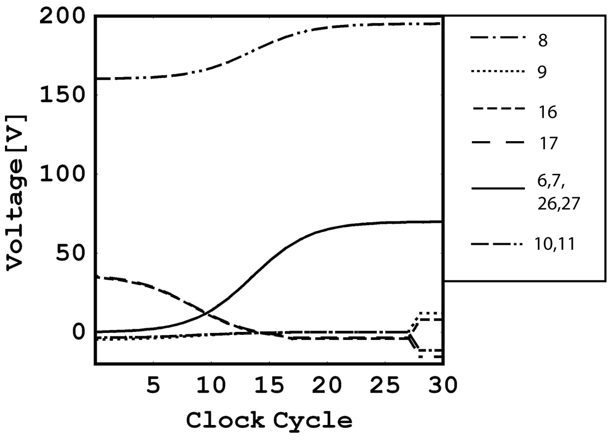

Fig. 4.9: Plot of time voltage profile of salient electrodes to shuttle the ion from the top of the T to the stem of the T.

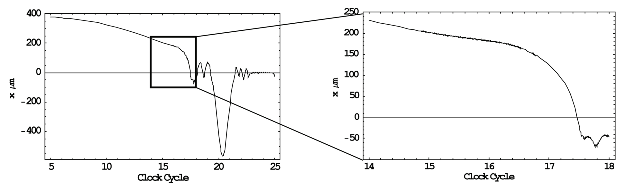

Fig. 4.10: Plot of x component of ion's motion as it is shuttled from the top to the Step. The zoom in shows the x motion as the ion is pushed up against the hump. Again, the position of the rf hump can be seen by the dramatic increase in micro-motion of the ion once the ion reaches the point $x = 220\mu m$. After the ion crosses the RF hump barrier, it rolls down the potential hill. The ion then oscillates wildly between regions d, e and f. Finally, the ion is caught by the t_{catch} step. (Not shown in Figure)

the ion failed to turn the corner. Refinements to the voltage sequence may allow for faster shuttling times.

Running the shuttling protocol for various values of t_{catch}, we find that t_{catch} can range between 25 to 42.

4.2.2 Time Reversibility of Shuttling Protocols

The shuttling protocol that had the most success shuttling the ion from i to d was a spatial reflection of the shuttling protocol from d to i and not a time reversal. The time reversal of the shuttling protocol failed at every attempt and variations of this time reversal yielded a success rate of at best 60%. As Newton's equations are time reversible, we would initially expect a time reversed shuttling protocol to work just as well as the original shuttling protocol. Therefore, the low success rate of this protocol is surprising.

However, once we take into account initial conditions, the apparent discrepancy is resolved. When starting the corner-shuttling protocol, the ion is laser cooled and therefore is at rest at a potential minimum. However, if we are to shuttle the ion from i to d via a time reversed voltage sequence, the initial conditions must match the ending conditions of the ion after being shuttled from d to i. Therefore, we must introduce a velocity in the starting condition or equivalently a displacement from the initial trap potential minimum. Obviously this is not possible and therefore, simply time reversing a voltage protocol will not in general yield a successful shuttling protocol.

4.3 Composite Protocols For Arbitrary Two Dimensional Control of Trapped Ions

With corner turning, linear shuttling and separation and recombination, we now have all the modules to create any arbitrary shuttling sequence in a 2 dimensional array of ion traps. We therefore experimentally demonstrate this by swapping the position of two ions in the T-trap.

We begin by trapping two different Cadmium isotopes in trapping zone *d*. The ion who's resonance is nearer the detection laser frequency is denoted ion B and thus appears brighter than ion A. We may therefore distinguish the two ions. This corresponds to the panel "Starting point" in Fig.4.11.

Next, we linear shuttle the two ions into trapping zone *b* and separate the ions such that ion A is in zone *a* and ion B is in zone *c*. We linear shuttle ion A into zone *d* and this corresponds to "Step 1" in Fig.4.11.

Ion B then turns the corner into region *i*. This corresponds to "Step 2" in Fig.4.11.

At "step 3". We linear shuttle Ion B from *i* to *k*. Then linear shuttle Ion A from *a* to *d*. Note, we cannot do both linear shuttles simultaneously since electrode 8 is required for both shuttling processes.

Ion A now turns the corner into region *i*. We then linear shuttle Ion A into region *h*. This process is equivalent to turning the corner from *d* to *f* and then linear shuttling from *f* to *h*, however, due to the possible electrode misalignment or a stray field, we cannot do this process in the more direct manner. After this, we linear shuttle Ion B from *k* to *i*. This corresponds to "Step 4" in Fig.4.11.

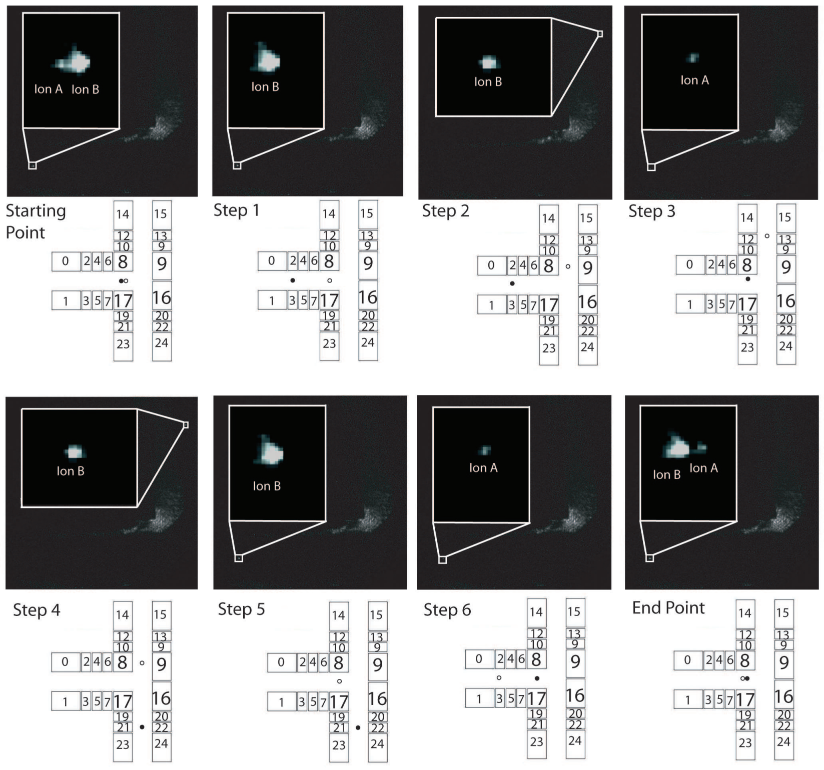

Fig. 4.11: Panels show video capture of swapping sequence. The swapping protocol is divided into a series of 8 steps corresponding to the eight panels in the figure. Each step corresponds to a point in time in the shuttling sequence where there is at least one ion in zones d or i. Every step consists of one or more shuttling modules. The shuttling modules within each step are run automatically without any time lag. On the other hand, each step is initialized by the experimenter. There is therefore, a short time lag, 1s between each run step. This is to allow one or both ions sufficient time to be laser cooled before executing the next step. These pauses are important in the steps where corner shuttling takes place, i.e. steps 2, 4, 5, 6. Without these time lags, the ion has insufficient time to be laser cooled causing the corner shuttling protocols to be unreliable.

We then corner shuttle ion B from i to d. This is "Step 5" in Fig.4.11.

In "Step 6", we first linear shuttle Ion B into trapping zone a. Next we linear shuttle Ion A from h to f and then corner shuttle back to d.

Finally, we linear shuttle Ion A from d to c and recombine the two ions at region b. We then linear shuttle the two ions from b to d. We have now successfully swapped the position of two ions in a single trap.

Altogether, we need 43 modules to create the swapping sequence. (Linear Shuttling an ion between two adjacent zones is counted as one module. therefore, shuttling the ion from i to k consists of two modules.) This shows that composite protocols can grow in complexity very quickly.

5. CONCLUSION

In this thesis, we outlined general strategies which allowed us to demonstrate the efficient arbitrary control of atomic ions in a multidimensional trap was demonstrated in the 11-zone, two dimensional ion trap array. In particular, accurate and computationally efficient simulations of the ion's classical motion was crucial to the design of key shuttling protocols.

The arbitrary two dimensional control of ions could allow for an efficient method to entangle arbitrarily positioned ions. In addition, our results, particularly in controlling ions around a junction also opens the door to more topologically complicated ion trap quantum computer architectures.

Future work will attempt to experimentally characterize the acquired kinetic energy through the shuttling protocols and to optimize the voltage sequences to optimize efficiency and adiabaticity. In addition, future experiments will characterize the coherence of a qubit after the various shuttling operations.

BIBLIOGRAPHY

[1] D. Deutsch. Quantum theory, the Church-Turing Principle and the Universal Quantum Computer. *Proc. Roy. Soc. London A*, 400:97, 1983

[2] M. Nielsen, I. Chuang. Quantum Computation and Quantum Information Cambridge,2000

[3] P.W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer *SIAM Journal of Computation* 26(5):1484-1509, 1997

[4] Simulating physics with computers *International Journal of Theoretical Physics* 21:467,1982

[5] D. J. Wineland, C. Monroe, W. M. Itano, D. Leibfried, B. E. King, and D. M. Meekhof."Experimental Issues in Coherent Quantum-State Manipulation of Trapped Atomic Ions", *Journal of Research of the National Institute of Standards and Technology* 103, 259 (1998).

[6] C. Monroe, D. M. Meekhof, B. E. King, W. M. Itano, and D. J. Wineland. Demonstration of a Fundamental Quantum Logic Gate *Phys. Rev. Lett.* 75, 4714 (1995).

[7] L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. Sherwood, and I. L. Chuang. Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance. *Nature*, 414 pp. 883, 2001.

[8] K.-A. Brickman, P. C. Haljan, P. J. Lee, M. Acton, L. Deslauriers, and C. Monroe. Implementation of Grover's Quantum Search Algorithm in a Scalable System *Phys. Rev. A* 72, 050306(R) (2005).

[9] Quantum Computations with Cold Trapped Ions.*Phys. Rev. Lett.* 74, 40914094 (1995)

[10] D. Kielpinski, C. Monroe, and D. J. Wineland, Architecture for a large-scale ion-trap quantum computer. *Nature* 417, 709 (2002).

[11] H. Metcalf, P. Straten. Laser Cooling and Trapping. Springer (1999)

[12] L. Deslauriers, M. Acton, B. B. Blinov, K.-A. Brickman, P. C. Haljan, W. K. Hensinger, D. Hucul, S. Katnik, R. N. Kohn, P. J. Lee, M. A. Madsen, P. Maunz, D. L. Moehring, S. Olmschenk, D. Stick, and C. Monroe. Efficient Photoionization-Loading of Trapped Cadmium Ions with Ultrafast Pulses (submitted, 2006).

[13] J.D. Jackson. Classical Electrodynamics. 3rd ed. Pg. 39.

[14] W. Paul. Electromagnetic traps for charged and neutral particles, *Rev. Mod. Phys*, 62, 531,(1990).

[15] H. Dehlmet. Radiofrequency spectroscopy of stored ions.*Adv. At. Mol. Phys.* 3, 53 (1967)

[16] D. Hucul, M. Yeo, W.K. Hensinger, J. Rabchuk, and C. Monroe. On the Transport of Atomic Ions in a Multi-dimensional Ion Trap Array.*In progress*(2006)

[17] J Leader. Numerical Analysis and Scientific Computation. Addison Wesley (February 5, 2004)

[18] J. Stoer, R. Bulirsch. R. Bartels, and W. Gautschi .Introduction to Numerical Analysis. Springer; 3 edition (August 21, 2002)

[19] W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling. Numerical Recipes in C : The Art of Scientific Computing. Chapter 16 p722-726 2nd ed. Cambridge, England: Cambridge University Press

[20] W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling. Numerical Recipes in C : The Art of Scientific Computing. Chapter 3 p108-109 2nd ed. Cambridge, England: Cambridge University Press

[21] M. Leok, Private Correspondence

[22] W. H. Press, B. P. Flannery, S. A. Teukolsky, William T. Vetterling.Numerical Recipes in C : The Art of Scientific Computing. Chapter 16 p747-752 2nd ed. Cambridge, England: Cambridge University Press

[23] L.F. Shampine and C.W. Gear, A Users view of Solving Stiff Ordinary Differential Equations, *SIAM Review*, 21 (1979) pp. 1-17.

[24] W. H. Press, B. P. Flannery, S. A. Teukolsky, W.T. Vetterling. Numerical Recipes in C : The Art of Scientific Computing. Chapter 16 p734-746 2nd ed. Cambridge, England: Cambridge University Press

[25] M. A. Rowe, A. Ben-Kish, B. DeMarco, D. Leibfried, V. Meyer, J. Beall, J. Britton, J. Hughes, W. M. Itano, B. Jelenkovic, C. Langer, T. Rosenband, and D. J. Wineland, Transport of Quantum States And Separation Of Ions In a Dual RF Ion Trap. *Quantum Information and Computation* 2, 257-271 (2002).

[26] M.D. Barrett, J. Chiaverini, T. Schaetz, J. Britton, W. M. Itano, J.D. Jost, E. Knill, C. Langer, D. Leibfried, R. Ozeri, and D. J. Wineland. Deterministic quantum teleportation of atomic qubits.*Nature* 429, 737 (2004).

[27] J. Chiaverini, J. Britton, D. Leibfried, E. Knill, M. D. Barrett, R.B. Blakestad, W.M. Itano, J.D. Jost, C. Langer, R. Ozeri, T. Schaetz, and D.J. Wineland,Implementation of the Semiclassical Quantum Fourier Transform in a Scalable System *Science* 308, 997 (2005).

[28] J. Chiaverini, D. Leibfried, T. Schaetz, M. D. Barrett, R. B. Blakestad, J. Britton, W.M. Itano, J.D. Jost, E. Knill, C. Langer, R. Ozeri, and D.J. Wineland, Realization of quantum error correction.*Nature* 432, 602(2004).

[29] W.K. Hensinger, J Rabchuk, S. Olmschenk, D. Stick, D. Hucul, M. Yeo, M. Acton, L. Deslauriers, and C. Monroe.T-junction ion trap array for two-dimensional ion shuttling, storage, and manipulation. *App. Phys. Lett.* 88, 034101 (2006).

[30] Merzbacher. “Quantum Mechanics.” John Wiley and Sons. New York. 1961.

[31] D. Hucul. Operation of a Two-Dimensional Ion Trap Array for Scalable Quantum Computation *in progress* (2006)