The Dissertation Committee for Robert Christian Behn
certifies that this is the approved version of the following dissertation:

The Transition from Subsonic to Supersonic Cracks

Committee:

Michael Marder, Supervisor

Richard Hazeltine

Richard Matzner

Philip Morrison

Krishnaswamy Ravi-Chandar

The Transition from Subsonic to Supersonic Cracks

by

Robert Christian Behn, B.S. PHY.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2016

Dedicated to my parents Bob and Lynn Behn.

Acknowledgments

I wish to thank my advisor Prof. Michael Marder for his support at every step of my study, for his patience and knowledge. His guidance helped me complete my research and write this thesis. I could not have imagined having a better advisor for my Ph.D. work.

I thank the rest of my thesis committee: Prof. Hazeltine, Prof. Matzner, Prof. Morrison, and Prof. Ravi-Chandar for their review of my work.

I also thank my parents for their continuous support throughout my graduate education.

The Transition from Subsonic to Supersonic Cracks

Publication No. ______________

Robert Christian Behn, Ph.D.
The University of Texas at Austin, 2016

Supervisor: Michael Marder

We present the full analytical solution for steady-state in-plane crack motion in a brittle triangular lattice. This allows quick numerical evaluation of solutions for very large systems, facilitating comparisons with continuum fracture theory. Cracks that propagate faster than the Rayleigh wave speed have been thought to be forbidden in the continuum theory, but clearly exist in lattice systems. Using our analytical methods, we examine in detail the motion of atoms around a crack tip as crack speed changes from subsonic to supersonic.

Subsonic cracks feature displacement fields consistent with a stress intensity factor. For supersonic cracks, the stress intensity factor disappears. Subsonic cracks are characterized by small-amplitude, high-frequency oscillations in the vertical displacement of an atom along the crack line, while supersonic cracks have large-amplitude, low-frequency oscillations. Thus, while supersonic cracks are no less physical than subsonic cracks, the connection between microscopic and macroscopic behavior must be made in a different way. This is one reason supersonic cracks in tension had been thought not to exist.

In continuum fracture theory, the energy flowing into the crack tip becomes negative or imaginary for crack speeds faster than the Rayleigh wave speed. This would suggest that supersonic cracks are not physically allowed. In response to this, we study the energy flow in our supersonic solutions in the lattice. First, we construct an energy flux vector in the lattice analogous to the Poynting vector in electromagnetism. This allows us to calculate the energy flow at each atom in the lattice. We find that there is positive energy flux into the crack tip for both subsonic and supersonic solutions in the lattice.

Table of Contents

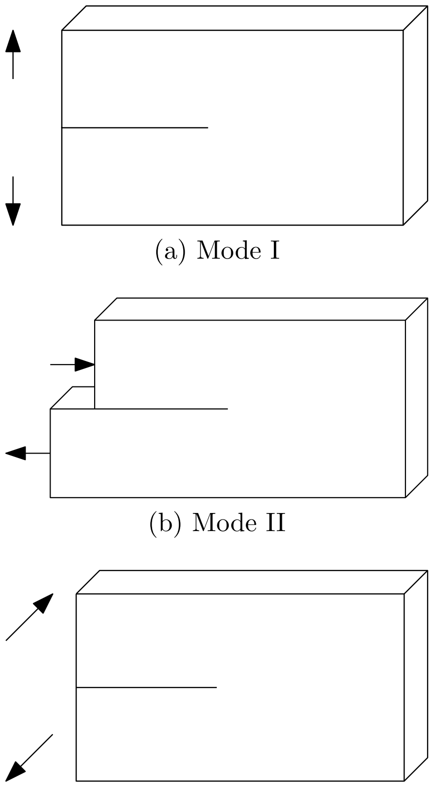
List of Figures

Chapter 1

Introduction

Fracture is a phenomenon from everyday life that everyone is familiar with. The study of the physics of fracture yields many interesting and unexpected results. For example, throwing a brick at a pane of glass creates a few atomically thin cracks that run at the speed of sound from end to end.

Here we present some results from two theories of fracture mechanics. The first theory takes place in a continuum, where a fracturing material is treated as a solid and the motion of a crack can be understood using equations of linear elasticity. The second is an atomic theory, taking place in a lattice, in which exact analytical calculations of crack motion can be performed for arbitrarily large systems. The advantage of the atomic theory is that the mathematical starting point is simple and unambiguous, and many analytical solutions are possible. In this work, we present original results in the atomic theory of fraction mechanics and attempt to reconcile some differences with the continuum theory.



(a) Mode I

(b) Mode II

(c) Mode III

Figure 1.1: Different fracture modes

1.1 Continuum background

When studying the motion of a crack, whether in a continuum or a lattice, we focus on three symmetrical configurations called modes. These are shown in Figure 1.1. There are mixed-mode situations that occur from certain combinations of forces, but for the study of the physics of cracks we restrict our attention to these three modes with high degrees of symmetry. For Mode I, which is the focus of this work, the crack faces, under tension, are displaced in a direction normal to the fracture plane. In Mode II, the motion is along the fracture plane. Mode III is an out of plane tearing where the motion is normal to the plane of the material.

The development of continuum fracture mechanics is very elaborate. There is a formalism developed by Muskhelishvili [23] that uses conformal mapping to compute the stress fields around two-dimensional static cracks in a solid. Here, we present this technique modified to find the fields around a moving crack[1]. We find the displacements and stress around a Mode I crack, and use these results to compute the energy flowing into the crack tip as a function of the crack velocity.

We start with the equation of motion for an isotropic elastic body in a continuum

$$\rho \frac{\partial^2 \mathbf{u}}{\partial t^2} = \mu \nabla^2 \mathbf{u} + (\lambda + \mu) \nabla (\nabla \cdot \mathbf{u}). \tag{1.1}$$

This is the Navier-Cauchy equation. The vector $\mathbf{u}$ is a field describing the dis-

[1] M. Marder, notes on fracture

placement of each mass point from its starting location in an unstrained body, ρ is the density, and the constants μ and λ are Lamé constants, which have dimensions of energy per volume, and are usually on the order 10^{10} ergs/cm^3. We can now find the solution for a Mode I crack. Consider Eq. (1.1) applied to a steady state in a moving frame with $x = vt$. We get

$$\rho v^2 \frac{\partial^2 \mathbf{u}}{\partial x^2} = \mu \nabla^2 \mathbf{u} + (\lambda + \mu)\nabla(\nabla \cdot \mathbf{u}). \tag{1.2}$$

We decompose $\mathbf{u}$ into transverse and longitudinal parts

$$\mathbf{u} = \mathbf{u}_t + \mathbf{u}_l \tag{1.3}$$

which are defined in terms of potentials v_t and v_l as

$$\mathbf{u}_l = \nabla v_l, \quad \mathbf{u}_t = \left(\frac{\partial v_t}{\partial y}, -\frac{\partial v_t}{\partial x} \right). \tag{1.4}$$

It follows that

$$\left[(\lambda + 2\mu)\nabla^2 - \rho v^2 \frac{\partial^2}{\partial x^2} \right] \mathbf{u}_l = \mathbf{f} = \left[\mu \nabla^2 - \rho v^2 \frac{\partial^2}{\partial x^2} \right] \mathbf{u}_t \tag{1.5}$$

for some harmonic function $\mathbf{f}$ (where $f_x - i f_y$ is a function of $x + iy$). We get

$$\left[\alpha^2 \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right] \nabla^2 v_l = 0 \tag{1.6}$$

$$\left[\beta^2 \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right] \nabla^2 v_t = 0 \tag{1.7}$$

where

$$\alpha^2 = 1 - \frac{\rho v^2}{\lambda + 2\mu} \tag{1.8a}$$

$$\beta^2 = 1 - \frac{\rho v^2}{\mu}. \tag{1.8b}$$

The general form of the potentials is

$$v_l = v_l^0(z) + \overline{v_l^0(z)} + v_l^1(x + i\alpha y) + \overline{v_l^1(x + i\alpha y)} \tag{1.9}$$

$$v_t = v_t^0(z) + \overline{v_t^0(z)} + v_t^1(x + i\beta y) + \overline{v_t^1(x + i\beta y)} \tag{1.10}$$

with Eq. (1.5) giving the relation between v_l^0 and v_t^0. These harmonic functions vanish from the expressions for $\mathbf{u}$ and can be neglected. We define $\phi(z) = \partial v_l^1(z)/\partial z$ and $\psi(z) = \partial v_t^1(z)/\partial z$. We can now write the components of $\mathbf{u}$ as

$$u_x = \phi(z_\alpha) + \overline{\phi(z_\alpha)} + i\beta \left[\psi(z_\beta) - \overline{\psi(z_\beta)}\right] \tag{1.11a}$$

$$u_y = i\alpha \left[\phi(z_\alpha) + \overline{\phi(z_\alpha)}\right] - \left[\psi(z_\beta) + \overline{\psi(z_\beta)}\right] \tag{1.11b}$$

where

$$z_\alpha = x + i\alpha y, \quad z_\beta = x + i\beta y. \tag{1.12}$$

Equation (1.11) gives the general solution for linear elasticity problems in a steady state moving at a constant velocity v. The components of the stress tensor are given by

$$\sigma_{ij} = \lambda\delta_{ij}u_{k,k} + \mu(u_{i,j} + u_{j,i}). \tag{1.13}$$

Let us define $\Phi(z) = \partial\phi(z)/\partial z$ and $\Psi(z) = \partial\psi(z)/\partial z$. The stresses are given by

$$\sigma_{xx} + \sigma_{yy} = 2(\lambda + \mu) \left[\Phi(z_\alpha) + \overline{\Phi(z_\alpha)}\right] (1 - \alpha^2) \tag{1.14a}$$

$$\sigma_{xx} - \sigma_{yy} = 2\mu \left\{(1 + \alpha^2) \left[\Phi(z_\alpha) + \overline{\Phi(z_\alpha)}\right] + 2i\beta \left[\Psi(z_\beta) - \overline{\Psi(z_\beta)}\right]\right\} \tag{1.14b}$$

$$2\sigma_{xy} = 2\mu \left\{2i\alpha \left[\Phi(z_\alpha) - \overline{\Phi(z_\alpha)}\right] - (\beta^2 + 1) \left[\Psi(z_\beta) + \overline{\Psi(z_\beta)}\right]\right\}. \tag{1.14c}$$

We also note that

$$\frac{\partial u_y}{\partial x} - \frac{\partial u_x}{\partial y} = -(1-\beta^2)\left[\Psi(z_\beta) + \overline{\Psi(z_\beta)}\right]. \tag{1.15}$$

The stresses are given directly by

$$\sigma_{xx} = \mu(1+2\alpha^2-\beta^2)\left[\Phi(z_\alpha) + \overline{\Phi(z_\alpha)}\right] + 2i\beta\mu\left[\Psi(z_\beta) - \overline{\Psi(z_\beta)}\right] \tag{1.16}$$

$$\sigma_{yy} = -\mu(1+\beta^2)\left[\Phi(z_\alpha) + \overline{\Phi(z_\alpha)}\right] - 2i\beta\mu\left[\Psi(z_\beta) - \overline{\Psi(z_\beta)}\right]. \tag{1.17}$$

We can solve a general problem by finding functions ϕ and ψ that match boundary conditions. Let us turn to the problem of a crack moving under symmetric loading at a constant speed v. Assume the crack lies along the negative x-axis. The crack tip is at $x = 0$ and moves in the positive x direction. We only need to assume the problem is symmetric under reflection about the x-axis.

We know in the case of a static crack that the stress fields have a square root singularity at the crack tip. We will assume this to be true in this problem as well, as it is true in all cases that can be worked explicitly. Near the crack tip, we can write

$$\phi(z) \sim (A_r + iA_i)z^{1/2} \tag{1.18}$$

$$\psi(z) \sim (B_r + iB_i)z^{1/2}. \tag{1.19}$$

The symmetry of the problem gives

$$u_x(-y) = u_x(y), \quad u_y(-y) = -u(y). \tag{1.20}$$

Plugging Eqs. (1.18) and (1.19) into Eq. (1.11) and using the symmetries in Eq. (1.20), we find that $A_i = B_r = 0$, therefore

$$\Phi(z) \sim \frac{A_r}{z^{1/2}}, \quad \Psi(z) \sim \frac{iB_i}{z^{1/2}}. \tag{1.21}$$

The square roots in the expressions of the potentials Eqs. (1.18) and (1.19) have their branch cuts along the negative x-axis corresponding to the crack. On the crack surface, we require that the stresses σ_{xy} and σ_{yy} vanish. Substituting Eq. (1.21) into Eq. (1.17), we find that $\sigma_{yy} = 0$ is satisfied for $x < 0$ and $y = 0$. Substituting into Eq. (1.14c) with $y = 0$ gives

$$\sigma_{xy} = i\mu \left[2\alpha A_r - (\beta^2+1)B_i\right] \left\{\frac{1}{\sqrt{x}} - \frac{1}{\sqrt{x^*}}\right\} \tag{1.22}$$

therefore

$$\frac{B_i}{A_r} = \frac{2\alpha}{\beta^2+1}. \tag{1.23}$$

Here we used the $\overline{z} = z^*$ notation to denote the complex conjugate. The stresses rewritten using Eq. (1.23) are

$$\sigma_{xx} = \frac{K}{\sqrt{2\pi}D}\left[(\beta^2+1)(1+2\alpha^2-\beta^2)\left\{\frac{1}{\sqrt{z_\alpha}} + \frac{1}{\sqrt{z_\alpha^*}}\right\} - 4\alpha\beta\left\{\frac{1}{\sqrt{z_\beta}} + \frac{1}{\sqrt{z_\beta^*}}\right\}\right] \tag{1.24}$$

$$\sigma_{yy} = \frac{K}{\sqrt{2\pi}D}\left[-(\beta^2+1)\left\{\frac{1}{\sqrt{z_\alpha}} + \frac{1}{\sqrt{z_\alpha^*}}\right\} + 4\alpha\beta\left\{\frac{1}{\sqrt{z_\beta}} + \frac{1}{\sqrt{z_\beta^*}}\right\}\right] \tag{1.25}$$

$$\sigma_{xy} = \frac{K}{\sqrt{2\pi}D}2i\alpha(\beta^2+1)\left\{\frac{1}{\sqrt{z_\alpha}} - \frac{1}{\sqrt{z_\alpha^*}} - \frac{1}{\sqrt{z_\beta}} + \frac{1}{\sqrt{z_\beta^*}}\right\} \tag{1.26}$$

with

$$D = 4\alpha\beta - (1+\beta^2)^2. \tag{1.27}$$

The constant K is the called the stress intensity factor and is given by

$$K = \lim_{x \to 0^+} \sqrt{2\pi x}\sigma_{yy}. \tag{1.28}$$

The energy flux in this problem can be found by taking the time derivative of the total energy (kinetic plus potential)

$$\frac{d}{dt}[K + P] = \frac{d}{dt}\int dxdy \, \left[\frac{\rho}{2}\dot{u}_\alpha \dot{u}_\alpha + \frac{1}{2}\frac{\partial u_\alpha}{\partial x_\beta}\sigma_{\alpha\beta}\right]. \tag{1.29}$$

The spatial integration is taken over an area that is static in the laboratory frame. The stress tensor is symmetric so we get

$$\frac{d}{dt}[K + P] = \int dxdy \, \left[\rho \ddot{u}_\alpha \dot{u}_\alpha + \frac{\partial \dot{u}_\alpha}{\partial x_\beta}\sigma_{\alpha\beta}\right]. \tag{1.30}$$

Inserting the equation of motion

$$\rho \ddot{u}_\alpha(\mathbf{x}) = \frac{\partial}{\partial x_\beta}\sigma_{\alpha\beta}(\mathbf{x}) \tag{1.31}$$

we can write the energy flux as

$$\begin{aligned} J^{\text{tot}} &= \int dxdy \, \left[\frac{\partial}{\partial x_\beta}\sigma_{\alpha\beta}\dot{u}_\alpha + \frac{\partial \dot{u}_\alpha}{\partial x_\beta}\sigma_{\alpha\beta}\right] && (1.32)\\ &= \int dxdy \, \frac{\partial}{\partial x_\beta}\left[\sigma_{\alpha\beta}\dot{u}_\alpha\right] && (1.33)\\ &= \int_{\partial S} \dot{u}_\alpha \sigma_{\alpha\beta} n_\beta && (1.34) \end{aligned}$$

where the last integral is taken over the boundary of the area, and the n_β are the components of an outward unit vector normal to the boundary.

Using the asymptotic expression for σ_{yy} from Eq. (1.25) and the corresponding expressions for u_y from Eq. (1.11b), we find that the total energy flowing into the crack tip per unit time is

$$J^{\text{tot}} = v(1-\beta^2)\frac{\alpha}{2\mu}\frac{1}{4\alpha\beta-(1+\beta^2)^2}K^2 \tag{1.35}$$

where v is the crack speed, c_l and c_t are longitudinal and transverse wave speeds respectively, $\alpha = \sqrt{1-v^2/c_l^2}$, $\beta = \sqrt{1-v^2/c_t^2}$, μ is a Lamé constant, and K, the Mode I dynamic stress intensity factor, is the coefficient of a universal singularity that develops outside of cracks that run as they are pulled symmetrically in tension from above and below.

One of the conclusions frequently drawn from the continuum theory of fracture mechanics is that cracks in tension cannot travel faster than the speed at which sound travels over a flat surface, the Rayleigh wave speed [4, 10, 27]. The reason for this assertion is that the motion of a crack requires energy to break bonds, but energy flux seems to become nonsensical for supersonic cracks. In particular, the rate at which energy flows into a crack tip per time is given by J^{tot} from Eq. (1.35), The denominator of this expression vanishes when the crack speed v reaches the Rayleigh wave speed, and for slightly higher velocities it becomes negative. Once the crack speed exceeds the transverse wave speed, the expression becomes imaginary. An expression saying that cracks moving above the Rayleigh wave speed need negative energy seems physically impossible, and an expression requiring imaginary energy seems even worse. One resolution of these problems is to conclude that cracks traveling faster than the Rayleigh wave speed in tension are not physically allowed.

1.2 Supersonic crack evidence

However, there is evidence that such cracks exist after all. The first indications came from measurements of earthquakes [14]. These suggested computer simulations [1, 11], and laboratory experiments [28] for cracks that move by sliding faces past each other, showing that cracks in shear (Mode II) can move faster than the transverse wave speed c_t (and hence the Rayleigh wave speed as well) and reach speeds close to the longitudinal speed c_l. Dynamic fracture theory was extended to include these "intersonic" cracks [4, 6, 26].

Cracks in rubber under tension were found to have a wedge-like tip suggestive of supersonic motion [8]. Additional experimental work confirmed that the cracks do travel faster than the transverse wave speed [7, 25]. The experiments led to theoretical descriptions for supersonic cracks in tension [19, 20]. Both explicit numerical solutions for atomic equations of motion and the corresponding analytical solutions show the supersonic cracks do in fact exist [13]. This last reference contains an extended discussion of how crack speed depends upon loading for the models studied in this work.

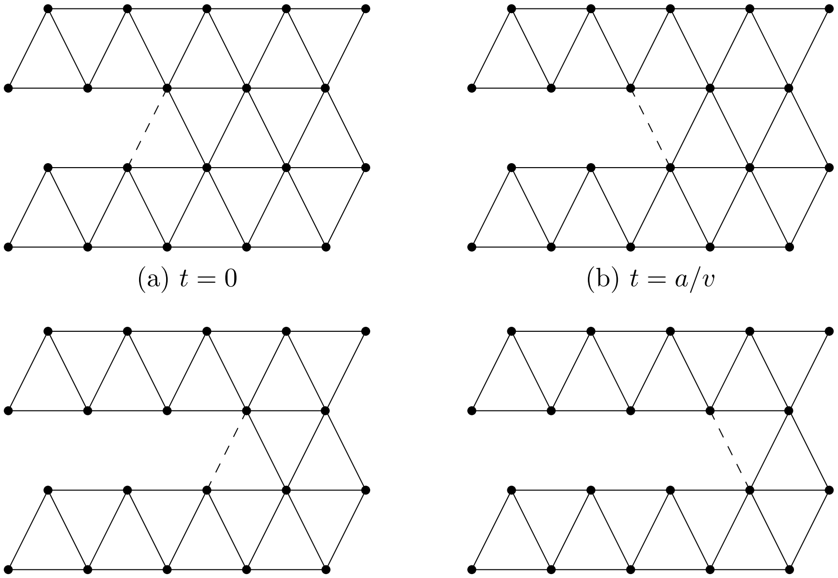
Supersonic cracks were also observed for cracks with hyperelastic constitutive laws that cause stiffening near the tip [5]. As shown, for example, in this work supersonic cracks exist equally well in materials without hyperelastic constitutive laws. Thus we are not sure that hyperelasticity is actually needed to promote supersonic cracks.

1.3 Atomic theory motivation

Our main technical tool to provide this discussion is the atomic theory of dynamic cracks. Analytical solutions at the atomic level for moving cracks were first found by Slepyan. His original calculations applied to out-of-plane (Mode III) cracks in an infinite square lattice [31]. He then generalized the solutions to in-plane (Mode I) cracks in an infinite square lattice [16]. Many additional solutions due to Slepyan are found in [30].

Additional solutions for cracks in a finite strip formed from a triangular lattice were found by Marder and Gross [22]. This paper also considers linear and nonlinear instabilities of the solutions. Many additional observations on the experimental implications of the solutions in discrete lattices are contained in [18]. Figure 1.2 shows a crack moving at a constant speed v in a triangular lattice with spacing a. The bonds between atoms along the crack line break at time intervals of $\Delta t = a/v$. This is the lattice studied in the paper [22] and expanded upon in this work.

In all these analytical solutions, the central quantity obtained from the analytical methods is a relationship between the external load applied to make a crack move, and the speed v at which the crack travels. In fact, the speed v of the crack enters the theory as an input parameter, and the calculations give the load as an output. For subsonic cracks, one can equivalently say that one obtains $K_I(v)$, the stress intensity factor as a function of crack speed. For supersonic cracks, the end result is the external system strain e_{yy} as a function of crack speed [20].



(a) $t = 0$

(b) $t = a/v$

(c) $t = 2a/v$

(d) $t = 3a/v$

Figure 1.2: Snapshots of the motion of a crack tip showing the sequence of bond breaks as the crack advances. The lattice spacing is a and the crack is moving at a constant speed v. Bonds break at time intervals of $\Delta t = a/v$

The theory provides more than just this relation. In principle, the theory makes it possible to find the behavior of every atom as a function of time for a crack moving in steady state. This information is relatively easy to obtain for cracks moving in anti-plane shear [22]. However, for two-dimensional cracks where atoms move freely in the plane, the expressions are so lengthy that to our knowledge no one has completed the process of extracting analytically the motion of every atom.

1.4 One-dimensional model

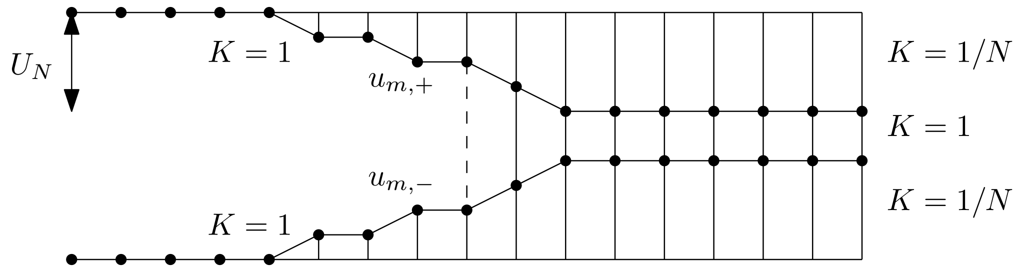


Figure 1.3: One-dimenional model.

We present here an example of these lattice calculations in the one-dimensional system shown in Figure 1.3, which can be viewed as a model for atoms lying along a crack surface [22]. The calculations for this system are relatively simple compared to the triangular lattice we work on in Chapter 2, but many of the equations we obtain hold for lattices.

The atoms in Figure 1.3 are attached to nearest neighbors in the same row by elastic springs with constant $K = 1$. They are attached to atoms on the other side of the crack line with the springs that snap when stretched past a breaking distance. The two rows of atoms are being pulled apart by weaker springs with constant $K = 1/N$, which are used to approximate N vertical rows of atoms pulling the lattice apart on each side. When we study the triangular lattice in Chapter 2, we calculate the motion of every atom in every row on the lattice.

The equation of motion for an atom on the upper row in this one-dimensional model is

$$\ddot{u}_{m,+} = \begin{array}{l} u_{m+1,+} - 2u_{m,+} + u_{m-1,+} \\ +(u_{m,-} - u_{m,+})\theta(2u_f - |u_{m,-} - u_{m,+}|) \\ +\frac{1}{N}(U_N - u_{m,+}) - b\dot{u}_{m,+} \end{array} \tag{1.36}$$

where θ is a step function of the stretch between atoms on opposite sides of the crack line, which vanishes when the stretch reaches a distance $2u_f$. The height of atoms after the crack has passed tends asymptotically to U_N, which is the boundary condition that drives the crack motion. There is a Stokes dissipation term with a small constant b, which turns out to have mathematical and physical importance.

As the system size N increases, the displacement U_N must also increase to continue driving the crack. We can find a dimensionless form of the boundary condition U_N. First, we need to find the minimum amount of energy stored in the strip, per lattice spacing far to the right of the crack tip, needed to break one bond along the crack line. The atoms far to the right of the crack tip are displaced a small amount U_{right} above the crack line. We define the ratio

$$Q_0 \equiv \frac{U_{\text{right}}}{U_N}. \tag{1.37}$$

We assume atoms far to the left and right of the crack are motionless and the dissipation is negligible. Far to the right, the atoms are displaced by $U_{\text{right}} = u_{m,+} = -u_{m,-}$, so balancing the spring forces gives

$$2U_{\text{right}} = \frac{1}{N}(U_N - U_{\text{right}}) \tag{1.38}$$

$$Q_0 = \frac{1}{2N+1}. \tag{1.39}$$

The energy per bond to the right is

$$E_{\text{right}} = 2\frac{1}{2}\frac{1}{N}(U_N - U_{\text{right}})^2 + \frac{1}{2}(2U_{\text{right}})^2 = 2Q_0(U_N)^2. \tag{1.40}$$

Far to the left of the crack, the displacement is

$$U_{\text{left}} = U_N. \tag{1.41}$$

The energy per bond to the left is that which is needed to snap a spring along the crack line and is given by

$$E_{\text{left}} = 2u_f^2. \tag{1.42}$$

Setting $E_{\text{left}} = E_{\text{right}}$ gives $2Q_0(U_N)^2 = 2u_f^2$, therefore the minimum value of the boundary condition U_N necessary to have enough energy stored to the right of the crack so bonds are able to snap along the crack line is

$$U_N^c = \frac{u_f}{\sqrt{Q_0}}. \tag{1.43}$$

We define the load Δ as a dimensionless measure of the displacement of the edges of the strip

$$\Delta \equiv \frac{U_N}{U_N^c} = \frac{U_N\sqrt{Q_0}}{u_f}. \tag{1.44}$$

Steady crack motion is only possible when $\Delta \geq 1$. In the case of $\Delta = 1$, all potential energy would have to go into snapping bonds for the crack to move.

Slepyan showed that atoms far from the crack tip are in motion when a crack moves in steady state [31], so the calculations above are based on false assumptions. It turns out that crack motion is only possible for $\Delta > 1$. The procedure to find the steady state solutions is as follows.

The steady state solutions have two important symmetries. First, there is a mirror symmetry with respect to the crack line. Second, the atom displacements repeat every $\Delta t = 1/v$, but shifted over one lattice spacing. These symmetries are written as

$$u_{m,+} = -u_{m,-} \tag{1.45}$$

$$u_{m,+}(t) = u_{0,+}(t - m/v) \tag{1.46}$$

where $u_{0,+}$ is the displacement of the atom at the crack tip on the upper row. This means all displacements along the lattice can be determined from the displacement of a single atom. Let us denote $u_{0,+}$ by $u(t)$. Applying Eqs. (1.45) and (1.46) to the equation of motion Eq. (1.36) gives

$$\ddot{u} = u(t - 1/v) - 2u + u(t + 1/v) + \frac{1}{N}(U_N - u) - 2u\theta(2u_f - 2|u|) - b\dot{u}. \tag{1.47}$$

Equation (1.47) can be solved analytically using the Wiener-Hopf technique [24]. Let us assume the bond breaks at $t = 0$, so $u(0) = u_f$ and the θ term vanishes afterwards. We also assume u increases to u_f this one time, and never falls back below it. We can now write

$$\ddot{u} = u(t - 1/v) - 2u + u(t + 1/v) + \frac{1}{N}\left(U_N e^{-\alpha|t|} - u\right) - 2u\theta(-t) - b\dot{u}. \tag{1.48}$$

Here we have introduced the factor $e^{-\alpha|t|}$ to avoid δ functions in the Fourier transforms. We will let α tend to zero at the end of the calculation. Before taking the Fourier transform of Eq. (1.48), we define

$$U^{\pm}(\omega) = \int dt\, e^{i\omega t}\theta(\pm t)u(t) \tag{1.49}$$

so that the Fourier transform of $u(t)$ is

$$U(\omega) = U^{+}(\omega) + U^{-}(\omega). \tag{1.50}$$

U^{+} is free of poles in the upper half complex ω plane, while U^{-} is free of poles in the lower half plane. This must be true for the integrals in Eq. (1.49) to be convergent. We can now take the Fourier transform of Eq. (1.48) to get

$$-\omega^2 U = 2(\cos(\omega/v)-1)+\frac{U_N}{N}\left\{\frac{1}{\alpha+i\omega}+\frac{1}{\alpha-i\omega}\right\}-\frac{U}{N}-2U^{-}+i\omega bU. \tag{1.51}$$

Rearranging gives

$$U(\omega)F(\omega) - 2U^{-}(\omega) = -\frac{U_N}{N}\left\{\frac{1}{\alpha+i\omega}+\frac{1}{\alpha-i\omega}\right\} \tag{1.52}$$

where

$$F(\omega) = \omega^2 + 2(\cos(\omega/v)-1) - \frac{1}{N} + i\omega b. \tag{1.53}$$

This can be rearranged using Eq. (1.50) to get

$$U^{+}(\omega)\frac{F(\omega)}{F(\omega)-2} + U^{-}(\omega) = -\frac{U_N}{N(F(\omega)-2)}\left\{\frac{1}{\alpha+i\omega}+\frac{1}{\alpha-i\omega}\right\}. \tag{1.54}$$

We define the function Q as

$$Q(\omega) = \frac{F(\omega)}{F(\omega)-2}. \tag{1.55}$$

As α tends to zero, the quantity in brackets in Eq. (1.54) becomes $\delta(\omega)$, so we can replace $F(\omega)$ in the denominator on the right hand side of the same equation with $F(0) = -1/N$ to get

$$Q(\omega)U^+(\omega) + U^-(\omega) = Q_0 U_N \left\{ \frac{1}{\alpha + i\omega} + \frac{1}{\alpha - i\omega} \right\} \tag{1.56}$$

with Q_0 given by Eq. (1.37) and $Q_0 = Q(0)$.

Eq. (1.56) is very important because the equations of motion of the lattice models we study can be reduced to this form and solved using the techniques presented in this section. The only difference is that the function Q becomes more complicated as the model becomes more realistic, which we will show in Chapter 2.

We can now apply the Wiener-Hopf technique to write

$$Q(\omega) = \frac{Q^-(\omega)}{Q^+(\omega)} \tag{1.57}$$

where Q^- is free of poles and zeroes in the lower complex ω plane and Q^+ is free of poles and zeroes in the upper complex plane, and these functions are given by

$$\begin{aligned} Q^{\pm}(\omega) &= \exp\left[\mp \int dt\, e^{i\omega t} \theta(\pm t) \int \frac{d\omega'}{2\pi} e^{-i\omega' t} \ln Q(\omega') \right] && (1.58) \\ &= \exp\left[\lim_{\epsilon \to 0} \int \frac{d\omega'}{2\pi} \frac{\ln Q(\omega')}{i\omega \mp \epsilon - i\omega'} \right]. && (1.59) \end{aligned}$$

Using Eq. (1.57), we can rearrange Eq. (1.56) to get

$$\frac{U^+(\omega)}{Q^+(\omega)} - \frac{Q_0 U_N}{Q^-(0)(-i\omega + \alpha)} = \frac{Q_0 U_N}{Q^-(0)(i\omega + \alpha)} - \frac{U^-(\omega)}{Q^-(\omega)} \tag{1.60}$$

where the left hand side is free of poles in the upper half plane, and the right hand side is free of poles in the lower half plane. For this to be true, both sides must separately equal a polynomial, which has to vanish for all ω or the inverse Fourier transforms of U^- and U^+ will have unphysical $\delta(t)$ terms. We now can write

$$U^{\pm}(\omega) = U_N \frac{Q_0 Q^{\pm}(\omega)}{Q^-(0)(\alpha \mp i\omega)}. \tag{1.61}$$

This completes the solution for $U(\omega)$. The numerical techniques we present in Chapter 3 to evaluate the displacement $u(t)$ in the triangular lattice can also be applied to this one-dimensional model.

We can also compute the load Δ as a function of the crack velocity v. This is done by checking that the bond along the crack line does in fact break at the right time, which mathematically speaking is the condition

$$u(t) = u_f \quad \text{at} \quad t = 0 \tag{1.62}$$

since the spring between atoms $u_{0,+}$ and $u_{0,-}$ is supposed to break at $t = 0$. We usually set $u_f = 1$ in our numerical calculations, so the only parameter left to determine is the boundary condition U_N. Equation (1.62) fixes its value, and therefore also the load Δ given in Eq. (1.44). Eq. (1.62) may be written as

$$\lim_{t \to 0^-} \int \frac{d\omega}{2\pi} e^{-i\omega t} U^-(\omega) = u_f. \tag{1.63}$$

We can evaluate this integral by inspection. For $t > 0$ we have

$$\int d\omega\, e^{-i\omega t} U^-(\omega) = 0 \tag{1.64}$$

and we know that any function decaying as $1/\omega$ for large ω has a step function discontinuity at the origin when Fourier transformed. Therefore Eqs. (1.63) and (1.61) give

$$u_f = U_N Q_0 \frac{Q^-(\infty)}{Q^-(0)}. \tag{1.65}$$

From the definition of Q in Eq. (1.55), it follows that $Q(\infty) = 1$, and from the definition of $Q^\pm$ in Eq. (1.59) we get

$$Q^-(\infty) = Q^+(\infty) = 1. \tag{1.66}$$

Using the definition of Δ in Eq. (1.44), and Eq. (1.65), we get an expression for the load

$$\Delta = \frac{Q^-(0)}{\sqrt{Q_0}}. \tag{1.67}$$

We can write this result solely in terms of $Q(\omega)$. Using Eq. (1.59) for Q^- and the fact that $Q(-\omega) = \overline{Q(\omega)}$, we get

$$\begin{aligned} Q^-(0) &= \exp\left[\int \frac{d\omega'}{2\pi}\frac{1}{2}\left[\frac{\ln Q(\omega')}{\epsilon - i\omega'} + \frac{\ln Q(-\omega')}{\epsilon + i\omega'}\right]\right] && (1.68) \\ &= \exp\left[\int \frac{d\omega'}{2\pi}\left[\frac{1}{-2i\omega'}\left\{\ln Q(\omega') - \overline{\ln Q(\omega')}\right\} + \frac{\epsilon}{\epsilon^2 + \omega'^2}\ln Q(0)\right]\right] && (1.69) \\ &= \sqrt{Q_0}\exp\left[-\int \frac{d\omega'}{2\pi}\frac{1}{2i\omega'}\left\{\ln Q(\omega') - \overline{\ln Q(\omega')}\right\}\right]. && (1.70) \end{aligned}$$

Finally, plugging Eq. (1.70) into Eq. (1.67) gives

$$\Delta = \exp\left[-\int \frac{d\omega'}{2\pi}\frac{1}{2i\omega'}\left\{\ln Q(\omega') - \overline{\ln Q(\omega')}\right\}\right]. \tag{1.71}$$

Eqs. (1.61) and (1.71) complete the formal solution of the one-dimensional model, giving the displacement along the crack line $u(t)$ and the load Δ necessary to drive the crack motion at a velocity v.

Figure 1.4 shows a plot of $\Delta(v)$ made using Eq. (1.71). All steady states occur for $\Delta > 1$, which means that not all energy stored to the right of the crack tip is used to snap bonds along the crack line. As the dissipation b tends to zero, the remaining energy is carried off by traveling waves as shown in Figure 1.5, which depicts a solution for $v = 0.5$, $N = 9$ and $b = 0.01$. For nonzero b, these traveling waves decay, and the extra energy is lost to dissipation.

Not all states along the Δ curve in Figure 1.4 are physical. It was shown by Marder [22] that states are linearly unstable when v decreases as Δ increases. This eliminates many of the states in the jagged lower portion of the curve. Another problem occurs for states in the range $0 < v < 0.3$ (the exact upper bound depends on b and N). They have the unphysical character shown in Figure 1.6. The atom rises above the height u_f before $t = 0$ and the bond does not snap. It then falls back down and the bond snaps at $t = 0$. This violates the original equation of motion Eq. (1.47), therefore states in this velocity range are unphysical.

1.5 Outline of this work

In this work, we have three aims. The first is to present for the first time analytical solutions for every atom in a two-dimensional lattice where a crack moves in plane. The algebra is extremely lengthy, and is presented in full in this work. Then, as an application of these exact solutions, we turn to the transition between subsonic and supersonic cracks. We examine

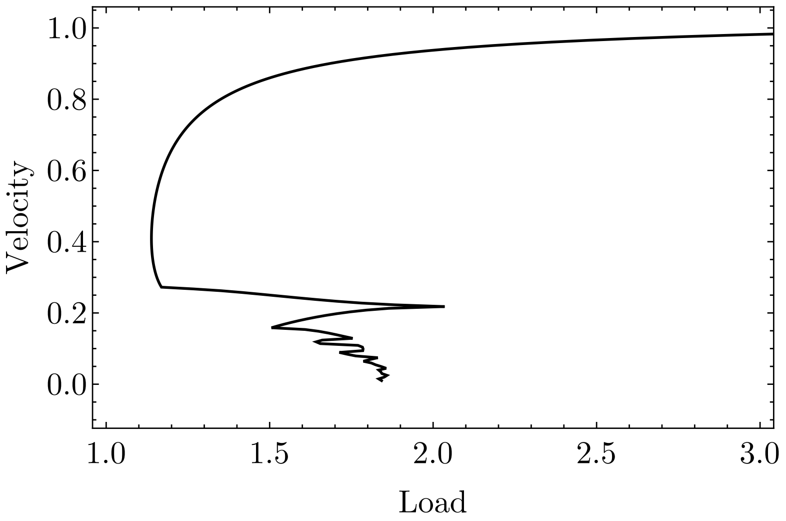


Figure 1.4: Crack velocity v versus load Δ for $N = 100$ and $b = 10^{-4}$.

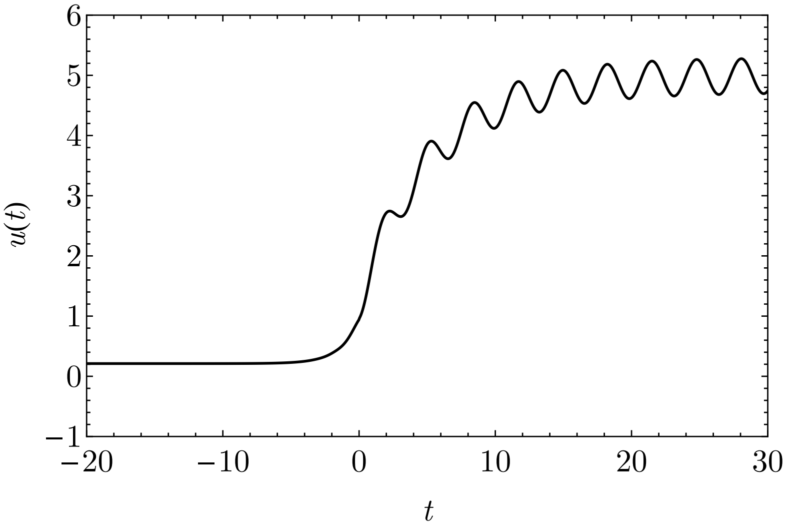


Figure 1.5: Height of $u_{0,+}$ versus time for $v = 0.5$, $N = 9$ and $b = 0.01$.

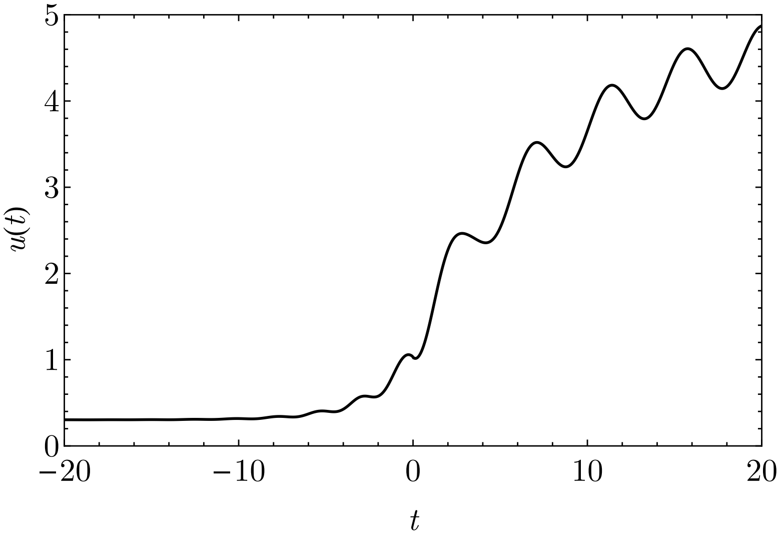


Figure 1.6: Height of $u_{0,+}$ versus time for $v = 0.3$, $N = 9$ and $b = 0.01$.

the precise way that the stress intensity factor vanishes across the transition point, and show what the cracks look like once they pass the Rayleigh wave speed. Because the methods produce solutions very rapidly, we are also able to discuss how these phenomena depend upon system size. Finally, we calculate the energy flux vector in the lattice and study its behavior for subsonic and supersonic cracks.

Chapter 2

Analytical in-plane solution

We present a description of the lattice used in our analysis of the in-plane crack, the starting equations, all analytical steps and finally the full algebraic expressions of the solution[1].

2.1 Lattice description

We carry out our computations in a triangular lattice of atoms with $2(N+1)$ rows. The motion of each atom is described by the displacement $\mathbf{u}_{m,n}$ from its equilibrium position, with m the column and n the row. The index m takes integer values while n takes values of the form $(k+1/2)$, with k an integer ranging from $-(N+1)$ to N.

The atoms in the lattice interact linearly with their nearest neighbors. The interaction is a function of both parallel and perpendicular displacements, with two spring constants $k_{\parallel}$ and $k_{\perp}$. We must make the restriction $k_{\perp}=0$ along the crack line, otherwise the mathematical formalism fails. We can make up for this by setting $k_{\parallel}=k_{\parallel}^{I}$ for the two rows along the crack. The

[1] This chapter uses material from the paper [3] by Chris Behn and Michael Marder, titled 'The transition from subsonic to supersonic cracks' and published in the journal *Phil. Trans. R. Soc. A* in 2015. Michael Marder edited the text.

bonds between atoms snap when they are displaced a distance $2u_f$, and the interaction becomes zero.

We note that in the long-wavelength limit, the motions of atoms in our lattice is described by isotropic continuum elasticity. In particular longitudinal and transverse wave speeds are given, for unit mass and lattice spacing a, by

$$c_l = \sqrt{\frac{3(3k_\parallel + k_\perp)a^2}{8}} \tag{2.1a}$$

$$c_t = \sqrt{\frac{3(k_\parallel + 3k_\perp)a^2}{8}}. \tag{2.1b}$$

In turn, this correspondence implies that crack motion in this lattice contains all results of dynamic linear elastic fracture mechanics as a special case.

To study crack motion, one must impose loading on the system. We do this by displacing the top row of atoms in our strip by a fixed vertical amount U_N. The crack consists in a separation between the middle rows of the lattice, where $n = \pm 1/2$. We study the system when the crack moves at constant speed, in a steady state. Steady crack motion at velocity v in the continuum means that elastic fields are functions of $x - vt$. For cracks in a lattice, one cannot employ this definition. However, there is a related symmetry, which is that in a lattice of spacing a

$$\mathbf{u}_{m,n}(t) = \mathbf{u}_{m+1,n}(t + a/v). \tag{2.2}$$

This means that what an atom at some height does now, its neighbor to the right will repeat exactly at a time a/v later. Employing this relation lets us eliminate the m index and relate the components of different columns in the lattice to each other by a simple time symmetry. In particular, since

$$\mathbf{u}_{m,n}(t) = \mathbf{u}_{0,n}(t - ma/v) \tag{2.3}$$

we will be able to drop the index m and set about finding $\mathbf{u}_n(t)$.

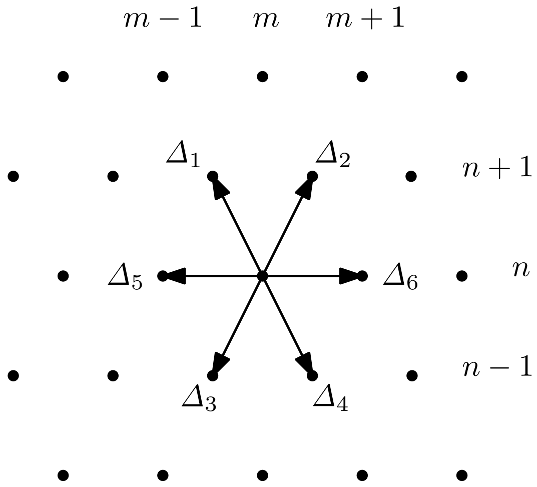


Figure 2.1: Diagram of the triangular lattice. Columns are indexed by m, rows are indexed by n. The $\mathbf{\Delta}_j$ are the six different displacement vectors.

2.2 Starting force equations

The lattice in which we solve for crack motion is shown in Figure 2.1. The displacement, for example, in the first direction is $\mathbf{\Delta}_1 = \mathbf{u}_{m-1,n+1} - \mathbf{u}_{m,n}$. To find the force on an atom, define $\mathbf{e}_{\|j}$ and $\mathbf{e}_{\perp j}$ to be unit vectors in the unstretched lattice. The unit vectors $\mathbf{e}_{\|j}$ point from atom 0 to atoms $j = 1 \ldots 6$ in Figure 2.1. Each of the unit vectors $\mathbf{e}_{\perp j}$ is perpendicular to the corresponding $\mathbf{e}_{\|j}$. That is,

$$\mathbf{e}_{\|j} \cdot \mathbf{e}_{\perp j} = 0. \tag{2.4}$$

The total force on any given atom, except those along the crack line where some bonds may be broken, is

$$\mathbf{F} = \sum_{j=1}^{6} \sum_{q=\|,\perp} k_q \mathbf{e}_{qj} \left(\mathbf{\Delta}_j \cdot \mathbf{e}_{qj} \right), \tag{2.5}$$

and the motion of atoms, of unit mass, is given by

$$\mathbf{F} = \ddot{\mathbf{u}} + b\dot{\mathbf{u}}. \tag{2.6}$$

The addition of Stokes dissipation through the second term on the right hand side is necessary to break time reversal invariance and tell the crack whether it should advance or retreat as time runs forward.

Define x to be the horizontal direction (the direction along which the crack moves) and y to be the vertical direction (along the width of the strip). The forces on atoms above the crack line, $n > 1/2$, take the explicit component form

$$F_n^x(t) = \begin{array}{l} \frac{1}{4}(k_\| + 3k_\perp)[u^x_{n+1}(t-(g_{n+1}-1)a/v) + u^x_{n+1}(t-g_{n+1}a/v) \\ +u^x_{n-1}(t-(g_{n-1}-1)a/v) + u^x_{n-1}(t-g_{n-1}a/v) - 4u^x_n(t)] \\ +\frac{\sqrt{3}}{4}(k_\perp - k_\|)[u^y_{n+1}(t-(g_{n+1}-1)a/v) - u^y_{n+1}(t-g_{n+1}a/v) \\ -u^y_{n-1}(t-(g_{n-1}-1)a/v) + u^y_{n-1}(t-g_{n-1}a/v)] \\ +k_\|[u^x_n(t+a/v) + u^x_n(t-a/v) - 2u^x_n(t)] \end{array} \tag{2.7}$$

$$F_n^y(t) = \begin{array}{l} \frac{1}{4}(k_\perp + 3k_\|)[u^y_{n+1}(t-(g_{n+1}-1)a/v) + u^y_{n+1}(t-g_{n+1}a/v) \\ +u^y_{n-1}(t-(g_{n-1}-1)a/v) + u^y_{n-1}(t-g_{n-1}a/v) - 4u^y_n(t)] \\ +\frac{\sqrt{3}}{4}(k_\perp - k_\|)[u^x_{n+1}(t-(g_{n+1}-1)a/v) - u^x_{n+1}(t-g_{n+1}a/v) \\ -u^x_{n-1}(t-(g_{n-1}-1)a/v) + u^x_{n-1}(t-g_{n-1}a/v)] \\ +k_\perp[u^y_n(t+a/v) + u^y_n(t-a/v) - 2u^y_n(t)]. \end{array} \tag{2.8}$$

The number g_n helps keep track of location on the triangular lattice.

It is defined by

$$g_n = \begin{cases} 0 & \text{if } n = 1/2, 5/2 \ldots \\ 1 & \text{if } n = 3/2, 7/2 \ldots, \\ \operatorname{mod}(n - 1/2, 2) & \text{in general.} \end{cases} \tag{2.9}$$

2.3 Analytical steps

Much of the algebra needed to solve this system appears in detail in [22]. There is a missing factor of 12 in the equations of motion (VI.21, p. 48) that appear in [22], although the results reported in the body of the paper are correct. The corrected forces along the crack line are

$$F^x = \begin{array}{l} \frac{1}{4}(k_\parallel + 3k_\perp)[u^x_{3/2}(t) + u^x_{3/2}(t - a/v) - 2u^x_{1/2}(t)] \\ +\frac{\sqrt{3}}{4}(k_\perp - k_\parallel)[u^y_{3/2}(t) - u^y_{3/2}(t - a/v)] \\ +k_\parallel[u^x_{1/2}(t + a/v) + u^x_{1/2}(t - a/v) - 2u^x_{1/2}(t)] \\ -\frac{\sqrt{3}}{2}k^I_\parallel[U(t)\theta(-t) - U(t - a/(2v))\theta(a/(2v) - t)] \end{array} \tag{2.10}$$

$$F^y = \begin{array}{l} \frac{1}{4}(k_\perp + 3k_\parallel)[u^y_{3/2}(t) + u^y_{3/2}(t - a/v) - 2u^y_{1/2}(t)] \\ +\frac{\sqrt{3}}{4}(k_\perp - k_\parallel)[u^x_{3/2}(t) - u^x_{3/2}(t - a/v)] \\ +k_\perp[u^y_{1/2}(t + a/v) + u^y_{1/2}(t - a/v) - 2u^y_{1/2}(t)] \\ -\frac{3}{2}k^I_\parallel[U(t)\theta(-t) - U(t - a/(2v))\theta(a/(2v) - t)] \end{array} \tag{2.11}$$

where $U(t)$ is a linear combination of $u^x(t)$ and $u^y(t)$ multiplying the Heaviside $\theta(t)$ functions:

$$U(t) = \frac{-1}{2\sqrt{3}}\left[u^x_{1/2}(t + a/(2v)) - u^x_{1/2}(t)\right] + \frac{1}{2}\left[u^y_{1/2}(t + a/(2v)) + u^y_{1/2}(t)\right]. \tag{2.12}$$

Fourier transforming the equations of motion gives

$$-(\omega^2 + ib\omega)u^x_{1/2}(\omega) = \begin{array}{l} \frac{1}{4}(k_\| + 3k_\perp)[u^x_{3/2}(\omega)(1 + e^{i\omega a/v}) - 2u^x_{1/2}(\omega)] \\ +\frac{\sqrt{3}}{4}(k_\perp - k_\|)[u^y_{3/2}(\omega)(1 - e^{i\omega a/v})] \\ +2k_\| u^x_{1/2}(\omega)[\cos(a\omega/v) - 1] \\ -\frac{\sqrt{3}}{2}k^I_\| U^-(\omega)(1 - e^{i\omega a/(2v)}) \end{array} \tag{2.13}$$

$$-(\omega^2 + ib\omega)u^y_{1/2}(\omega) = \begin{array}{l} \frac{1}{4}(k_\perp + 3k_\|)[u^y_{3/2}(\omega)(1 + e^{i\omega a/v}) - 2u^y_{1/2}(\omega)] \\ +\frac{\sqrt{3}}{4}(k_\perp - k_\|)[u^x_{3/2}(\omega)(1 - e^{i\omega a/v})] \\ +2k_\perp u^y_{1/2}(\omega)[\cos(a\omega/v) - 1] \\ -\frac{3}{2}k^I_\| U^-(\omega)(1 + e^{i\omega a/(2v)}) \end{array} \tag{2.14}$$

where

$$U^\pm(\omega) = \int dt\, e^{i\omega t}\theta(\pm t)U(t). \tag{2.15}$$

These equations can be combined and rearranged to give finally

$$Q(\omega)U^+(\omega) + U^-(\omega) = Q_0 U_N \left\{ \frac{1}{\alpha + i\omega} + \frac{1}{\alpha - i\omega} \right\}. \tag{2.16}$$

Here the function $Q(\omega)$ can be calculated explicitly from the Fourier transformed equations of motion for the lattice. We compute it from MAXIMA or Mathematica scripts, and for purposes of rapid numerical evaluation find it in Fortran. In addition $Q_0 = Q(0)$, and α is a small positive constant introduced when rewriting the boundary condition as $U_N e^{-\alpha|t|}$. The Fourier transform of this function is easier to deal with than $\delta(\omega)$. We let α tend to zero at the end of the calculation. This is the same procedure as was used in the one-dimensional model in Section 1.4.

The Wiener-Hopf technique [24] lets us write:

$$Q(\omega) = \frac{Q^-(\omega)}{Q^+(\omega)} \tag{2.17}$$

where Q^- is analytic in the lower half-plane, and Q^+ is analytic in the upper half-plane. This decomposition is given by the explicit formula:

$$Q^{\pm}(\omega) = \exp\left[\mp \int dt\, e^{i\omega t}\theta(\pm t) \int \frac{d\omega'}{2\pi} e^{-i\omega' t} \ln Q(\omega')\right]. \tag{2.18}$$

We can split Eq. (2.16) into two pieces, one which is analytic only in the upper half-plane, and one which is analytic only in the lower half-plane:

$$\frac{U^+(\omega)}{Q^+(\omega)} - \frac{Q_0 U_N}{Q^-(0)(-i\omega + \alpha)} = \frac{Q_0 U_N}{Q^-(0)(i\omega + \alpha)} - \frac{U^-(\omega)}{Q^-(\omega)}. \tag{2.19}$$

The left and right hand sides of the above equation are analytic in opposite sections of the complex plane, therefore they must both equal a polynomial $P(\omega)$. We must have $P(\omega) = 0$, otherwise $U^-(t)$ and $U^+(t)$ will include unphysical $\delta(t)$ terms. This lets us write

$$U^{\pm}(\omega) = U_N \frac{Q_0 Q^{\pm}(\omega)}{Q^-(0)(\alpha \mp i\omega)}. \tag{2.20}$$

From the two functions $U^+(\omega)$ and $U^-(\omega)$ one can obtain the linear combination of horizontal and vertical motions defined in Eq. (2.12) from

$$U(\omega) = U^-(\omega) + U^+(\omega). \tag{2.21}$$

Finally, one can obtain the horizontal and vertical motion of any atom in the system from

$$u_n^x(\omega) = C_n^x(\omega) U(\omega) \tag{2.22}$$

$$u_n^y(\omega) = C_n^y(\omega) U(\omega) + U_N \frac{(n - 1/2)}{N} \delta(\omega). \tag{2.23}$$

The functions $C_n^x(\omega)$ and $C_n^y(\omega)$, like $Q(\omega)$ are very lengthy, but can be obtained from essentially straightforward algebra resulting from the equations of motion. Again, we obtain explicit expressions using symbolic algebra programs, and implement the results for rapid evaluation in Fortran.

2.4 Kelvin dissipation

The solution we have described so far includes Stokes dissipation $-b\dot{\mathbf{u}}$, which was present in Eq. (2.6). However, it is not physically realistic in the lattice. It acts as though atoms are embedded in some sort of ether that slows them down according to the magnitude of their velocity in some arbitrarily chosen reference frame. A better form of dissipation from the physical point of view is Kelvin dissipation, which produces a force opposing atomic motion according to the relative motion of adjacent atoms. To preserve generality, we keep both Stokes and Kelvin dissipation in the equations of motion. The modification of our equations to include Kelvin dissipation is as follows. Suppose we have already solved

$$\ddot{\vec{u}}_i = L_i \left\{ \vec{u} \right\}, \tag{2.24}$$

where L_i is the operator producing the acceleration $\ddot{\vec{u}}_i$ after acting on all positions $\vec{u}$. The Fourier transform of this equation is

$$-\omega^2 \vec{U}_i = \tilde{L}_i \left\{ \vec{U} \right\}, \tag{2.25}$$

where $U(\omega)$ is the Fourier transform of $u(t)$. If we want to include Kelvin dissipation, we can almost write

$$\ddot{\vec{u}}_i = \left(1 + \beta \frac{\partial}{\partial t} \right) L_i \left\{ \vec{u} \right\}. \tag{2.26}$$

This expression is not quite right in that when bonds break the time derivative produces unphysical discontinuities on the right hand side. A solution of this problem was laid out in [21]. The solution is to define

$$w(t) \equiv \left(1 + \beta \frac{\partial}{\partial t}\right) u(t). \tag{2.27}$$

The correct equation to solve is

$$\ddot{\vec{u}}_i = L_i \left\{\vec{w}\right\} \tag{2.28}$$

so that the θ functions multiply displacements after the time derivative has been taken. The Fourier transform is

$$W(\omega) = \int dt\, e^{i\omega t} w(t) = (1 - i\beta\omega) U(\omega), \tag{2.29}$$

and therefore we get

$$-\frac{\omega^2}{1 - i\beta\omega} \vec{W}_i = \tilde{L}_i \left\{\vec{W}\right\}. \tag{2.30}$$

The equation for W is then given by making the following substitution in the equation for U:

$$\omega^2 \to \frac{\omega^2}{1 - i\beta\omega}. \tag{2.31}$$

The equations of motion on the crack line are then modified to read

$$-(\omega^2 + ib\omega) u^x_{1/2}(\omega) = \begin{array}{l} (1 - i\omega\beta)\, \frac{1}{4}(k_\parallel + 3k_\perp)[u^x_{3/2}(\omega)(1 + e^{i\omega a/v}) - 2u^x_{1/2}(\omega)] \\ + (1 - i\omega\beta)\, \frac{\sqrt{3}}{4}(k_\perp - k_\parallel)[u^y_{3/2}(\omega)(1 - e^{i\omega a/v})] \\ + (1 - i\omega\beta)\, 2k_\parallel u^x_{1/2}(\omega)[\cos(a\omega/v) - 1] \\ -\frac{\sqrt{3}}{2} k^I_\parallel W^-(\omega)(1 - e^{i\omega a/(2v)}) \end{array} \tag{2.32}$$

$$
\begin{aligned}
-(\omega^2 + ib\omega)u^y_{1/2}(\omega) = \; & (1 - i\omega\beta)\,\tfrac{1}{4}(k_\perp + 3k_\parallel)[u^y_{3/2}(\omega)(1 + e^{i\omega a/v}) - 2u^y_{1/2}(\omega)] \\
& + (1 - i\omega\beta)\,\tfrac{\sqrt{3}}{4}(k_\perp - k_\parallel)[u^x_{3/2}(\omega)(1 - e^{i\omega a/v})] \\
& + (1 - i\omega\beta)\,2k_\perp u^y_{1/2}(\omega)[\cos(a\omega/v) - 1] \\
& - \tfrac{3}{2}k^I_\parallel W^-(\omega)(1 + e^{i\omega a/(2v)}).
\end{aligned}
\tag{2.33}
$$

The functions $W^\pm$ are defined just like $U^\pm$ in Eq. (2.15). From this point the analysis proceeds as before, finding a new function $Q(\omega)$ that is not appreciably more complicated because of the presence of β. From this function one determines W using the Wiener-Hopf technique as

$$
W^\pm(\omega) = U_N \frac{Q_0 Q^\pm(\omega)}{Q^-(0)(\alpha \mp i\omega)}
\tag{2.34}
$$

$$
U(\omega) = \frac{W^-(\omega) + W^+(\omega)}{1 - i\beta\omega}.
\tag{2.35}
$$

Once $U(\omega)$ is in hand, the motion of every atom can again be determined. There is also a compact result relating the velocity v of a crack (which goes into the computation of Q) and the boundary extension U_N. In the triangular lattice, the boundary extension U_N can be expressed in terms of the dimensionless load Δ:

$$
U_N = 2u_f\Delta/\sqrt{3Q_0}.
\tag{2.36}
$$

Then, defining $\omega_0 = 1/\beta$, Δ can be computed from $Q(\omega)$ directly as

$$
\Delta = \exp\left[-\int \frac{d\omega}{4\pi}\left(\frac{\omega_0^2}{i\omega'(\omega_0^2 + \omega'^2)}[\ln Q(\omega') - \overline{\ln Q(\omega')}] + \frac{\omega_0 \ln|Q(\omega')|^2}{\omega_0^2 + \omega'^2}\right)\right].
\tag{2.37}
$$

This is a modification of the Δ from Section 1.4 to include Kelvin dissipation.

2.5 Full in-plane algebra

In Section 2.3, we wrote down the Fourier transformed equations of motion for atoms along the crack line. Here, we present the more general equations for atoms anywhere on the lattice and show the full algebra for the in-plane solution.

First, we apply the Fourier transform to the components above the crack line in Eqs. (2.7) and (2.8) to get

$$F_n^x(\omega) = \begin{array}{l} \frac{1}{4}(k_\parallel + 3k_\perp)[u_{n+1}^x(\omega)(e^{i\omega(g_{n+1}-1)a/v} + e^{i\omega g_{n+1}a/v}) \\ +u_{n-1}^x(\omega)(e^{i\omega(g_{n-1}-1)a/v} + e^{i\omega g_{n-1}a/v}) - 4u_n^x(\omega)] \\ +\frac{\sqrt{3}}{4}(k_\perp - k_\parallel)[u_{n+1}^y(\omega)(e^{i\omega(g_{n+1}-1)a/v} - e^{i\omega g_{n+1}a/v}) \\ -u_{n-1}^y(\omega)(e^{i\omega(g_{n-1}-1)a/v} - e^{i\omega g_{n-1}a/v})] \\ +k_\parallel u_n^x(\omega)(e^{i\omega a/v} + e^{-i\omega a/v} - 2) \end{array} \tag{2.38}$$

$$F_n^y(\omega) = \begin{array}{l} \frac{1}{4}(k_\perp + 3k_\parallel)[u_{n+1}^y(\omega)(e^{i\omega(g_{n+1}-1)a/v} + e^{i\omega g_{n+1}a/v}) \\ +u_{n-1}^y(\omega)(e^{i\omega(g_{n-1}-1)a/v} + e^{i\omega g_{n-1}a/v}) - 4u_n^y(\omega)] \\ +\frac{\sqrt{3}}{4}(k_\perp - k_\parallel)[u_{n+1}^x(\omega)(e^{i\omega(g_{n+1}-1)a/v} - e^{i\omega g_{n+1}a/v}) \\ -u_{n-1}^x(\omega)(e^{i\omega(g_{n-1}-1)a/v} - e^{i\omega g_{n-1}a/v})] \\ +k_\perp u_n^y(\omega)(e^{i\omega a/v} + e^{-i\omega a/v} - 2). \end{array} \tag{2.39}$$

Let

$$\begin{pmatrix} u_n^x \\ u_n^y \end{pmatrix} = y^n e^{i\omega g_n/(2v)} \begin{pmatrix} U_x \\ U_y \end{pmatrix}. \tag{2.40}$$

Substituting Eq. (2.40) into Eqs. (2.38) and (2.39) gives

$$-(m\omega^2 + i\omega b)U_x = \begin{array}{l} [(k_\parallel + 3k_\perp)\cos(\omega a/(2v))\frac{1}{2}(y + y^{-1}) \\ +2k_\parallel \cos(\omega a/v) - 3(k_\perp + k_\parallel)]U_x \\ -\sqrt{3}i(k_\perp - k_\parallel)\sin(\omega a/(2v))\frac{1}{2}(y - y^{-1})U_y \end{array} \tag{2.41}$$

$$-(m\omega^2 + i\omega b)U_y = \begin{array}{l} [(k_\perp + 3k_\parallel)\cos(\omega a/(2v))\frac{1}{2}(y + y^{-1}) \\ +2k_\perp \cos(\omega a/v) - 3(k_\perp + k_\parallel)]U_y \\ -\sqrt{3}i(k_\perp - k_\parallel)\sin(\omega a/(2v))\frac{1}{2}(y - y^{-1})U_x. \end{array} \tag{2.42}$$

We can solve for y by letting the determinant of this system vanish. First, write

$$z = \frac{1}{2}(y + y^{-1}). \tag{2.43}$$

The determinantal equation gives

$$\begin{aligned}
A &= 3(k_\perp - k_\|)^2 + 16 k_\perp k_\| \cos^2(\omega a/(2v)) && (2.44) \\
B &= \begin{array}{l} \cos(\omega a/(2v))[2(3k_{|||}^2 + k_\| k_\perp + 3k_\perp^2)\cos(\omega a/v) \\ +4(k_\perp + k_\|)(m\omega^2 + i\omega b - 3(k_\| + k_\perp))] \end{array} && (2.45) \\
C &= \begin{array}{l} [m\omega^2 + i\omega b - (k_\| + k_\perp)(3 - \cos(\omega a/v))]^2 \\ -(k_\| - k_\perp)^2[\cos^2(\omega a/v) + 3\sin^2(\omega a/(2v))] \end{array} && (2.46) \\
z_\pm &= \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}. && (2.47)
\end{aligned}$$

There are four values of y that satisfy Eqs. (2.43) and (2.47), two of which are

$$y_\pm = z_\pm + \sqrt{(z_\pm)^2 - 1}, \tag{2.48}$$

and the other two are given by the inverse of these, or subtracting the square root. Next, define

$$\begin{aligned}
D_\pm &= \begin{array}{l} m\omega^2 + i\omega b + (k_\| + 3k_\perp)\cos(\omega a/(2v))\frac{1}{2}(y_\pm + y_\pm^{-1}) \\ +2k_\| \cos(\omega a/v) - 3(k_\perp + k_\|) \end{array} && (2.49) \\
E_\pm &= -\sqrt{3}i(k_\perp - k_\|)\sin(\omega a/(2v))\tfrac{1}{2}(y_\pm - y_\pm^{-1}). && (2.50)
\end{aligned}$$

The general solution of Eqs. (2.38) and (2.39) is

$$\begin{aligned}
\begin{pmatrix} u_n^x \\ u_n^y \end{pmatrix} = & \; e^{-i\omega g_n a/(2v)} \begin{bmatrix} y_+^{n-1/2} \begin{pmatrix} E_+ \\ -D_+ \end{pmatrix} u_{1+} + y_+^{-n+1/2} \begin{pmatrix} E_+ \\ D_+ \end{pmatrix} u_{2+} \\ +y_-^{n-1/2} \begin{pmatrix} E_- \\ -D_- \end{pmatrix} u_{1-} + y_-^{-n+1/2} \begin{pmatrix} E_- \\ D_- \end{pmatrix} u_{2-} \end{bmatrix} \\
& +U_N \frac{(n-1/2)}{N} \begin{pmatrix} 0 \\ 1 \end{pmatrix}.
\end{aligned} \tag{2.51}$$

The four variables $u_{1\pm}$ and $u_{2\pm}$ are determined from the boundary conditions for the components

$$u^x_{N+1/2} = u^y_{N+1/2} - U_N = 0 \tag{2.52}$$

and Eq. (2.51) with $n = 1/2$

$$\begin{pmatrix} u^x_{1/2} \\ u^y_{1/2} \end{pmatrix} = \begin{pmatrix} E_+ \\ -D_+ \end{pmatrix} u_{1+} + \begin{pmatrix} E_+ \\ D_+ \end{pmatrix} u_{2+} \begin{pmatrix} E_- \\ -D_- \end{pmatrix} u_{1-} + \begin{pmatrix} E_- \\ D_- \end{pmatrix} u_{2-}. \tag{2.53}$$

Once $u_{1\pm}$ and $u_{2\pm}$ are found, we can use Eq. (2.51) to write $u^x_{3/2}$ and $u^y_{3/2}$ in terms of $u^x_{1/2}$ and $u^y_{1/2}$. These can be inserted into the crack line Eqs. (2.13) and (2.14), and we can then find expressions for $u^x_{1/2}(\omega)$ and $u^y_{1/2}(\omega)$ in terms of $U^-(\omega)$.

The section of the Fortran script used to calculate $Q(\omega)$ is given below. It shows every step and every expression used to numerically evaluate $Q(\omega)$, as well as the formulas for the components $u^x_n(\omega)$ and $u^y_n(\omega)$ in terms of $U^-(\omega)$. It is useful to see all the algebra to understand the complexity of the problem. The full script for calculating the time components $u^y_n(t)$ and $u^y_n(t)$ is shown in Appendix A.

```
diss=ii*w*b
w0=1.d0/bk
kdiss=1.d0/(1.d0-w/w0*ii)
eia2=exp(-a*ii*w/v/2.d0)
eia=eia2**2
eiai=1.d0/eia
eia2i=1.d0/eia2
caw=cos(a*w/v)
caw2=cos(0.5d0*a*w/v)
saw=sin(a*w/v)
```

```
 saw2=sin(0.5d0*a*w/v)
 kkr=0.75d0*kr+0.25d0*kp
 kkp=0.25d0*kr+0.75d0*kp
 krmkp=kr-1.d0*kp
 aa=16.d0*kp*kr*cos(0.5d0*a*w/v)**2+3.d0*(kr-1.d0*kp)**2
 bb=cos(0.5d0*a*w/v)*(2.d0*(3.d0*kr**2+
$  2.d0*kp*kr+3.d0*kp**2)*cos(a*w/v
$  )+4.d0*(kr+kp)*(m*w**2*kdiss+diss-3.d0*(kr+kp)))
 cc=((-1.d0*kr-1.d0*kp)*(3.d0-1.d0*
$  cos(a*w/v))+m*w**2*kdiss+diss)**2-
$  1.d0*(kp-1.d0*kr)**2*(cos(a*w/v)**2+3.d0*sin(0.5d0*a*w/v)**2)
 z_p=0.5d0*(sqrt(bb**2-4.d0*aa*cc)-1.d0*bb)/aa
 z_m=0.5d0*(-1.d0*sqrt(bb**2-4.d0*aa*cc)-1.d0*bb)/aa
 y_p=sqrt(z_p**2-1.d0)
 y_m=sqrt(z_m**2-1.d0)
 if(real(y_p)*real(z_p).gt.0.d0) then
    y_p=1.d0/(z_p+y_p)
 else
    y_p=1.d0/(z_p-y_p)
 end if
 if(real(y_m)*real(z_m).gt.0.d0) then
    y_m=1.d0/(z_m+y_m)
 else
    y_m=1.d0/(z_m-y_m)
 end if
 if(abs(y_p).gt.1.d0) y_p=1.d0/y_p
 if(abs(y_m).gt.1.d0) y_m=1.d0/y_m


 y_pn=y_p**n
 y_mn=y_m**n
 d_p=caw2*(3.d0*kr+kp)*z_p+
$  m*w**2*kdiss-3.d0*(kr+kp)+2.d0*caw*kp+diss
 d_m=caw2*(3.d0*kr+kp)*z_m+m*w**2*kdiss-
$  3.d0*(kr+kp)+2.d0*caw*kp+diss
 e_p=0.5d0*ii*(kp-1.d0*kr)*saw2*sqrt3*(y_p-1.d0/y_p)
 e_m=0.5d0*ii*(kp-1.d0*kr)*saw2*sqrt3*(y_m-1.d0/y_m)
 v1=d_p**2
 v2=e_m**2
 v3=v1*v2
 v4=-2.d0*d_m*d_p*e_m*e_p
 v5=d_m**2
 v6=e_p**2
 v7=v5*v6
 v8=-v1*v2
 v9=-d_m*d_p*e_m*e_p
 v10=y_m**(n+1)
 v11=-v5*v6
 v12=y_mn**2
 v13=y_pn**2
 v14=1.d0/((6.d0*d_m*d_p*e_m*e_p*y_m*y_mn+
$  2.d0*d_m*d_p*e_m*e_p*v10)*y_p*
$  y_pn+((v11+v9)*y_m*v12+(v9+v8)*v10*
$  y_mn+(v7+v4+v3)*y_m)*y_p+((v
$  7+v9)*y_m*v12+(v9+v3)*v10*y_mn+(v11+v4+v8)*y_m)*y_p*v13)
 v15=-eia2*v1*v2
 v16=d_m*d_p*eia2*e_m*e_p
 v17=y_m**2
 v18=y_m**n
 v19=d_m*d_p*eia2*e_m*e_p*v12
 v20=-eia2*v5*v6
 v21=d_m*d_p*eia2*e_m*e_p*v10*y_mn
 v22=y_p**2
 v23=-d_m*d_p*eia2*e_m*e_p*v10
 v24=-d_m*d_p*eia2*e_m*e_p*y_m*y_mn
 v25=v5*eia2*v6
 v26=v1*eia2*v2
 v27=d_p*eia2*v2*e_p
 v28=-d_m*eia2*e_m*v6
 v29=-d_p*eia2*e_p*v18*v2*y_mn
 v30=-d_p*eia2*e_p*v2
 v31=d_m*eia2*e_m*v6
 v32=d_p*eia2*v2*e_p*v10*y_mn
 v33=d_m*v1*eia2*e_m
 v34=-d_p*eia2*e_p*v5
 v35=v5*d_p*eia2*e_p*v12
 v36=-d_m*eia2*e_m*v1
 v37=v5*d_p*eia2*e_p
 v38=-d_p*eia2*e_p*v12*v5*y_m
 v39=-d_m*d_p*eia2*e_m*e_p
 v40=-d_m*d_p*eia2*e_m*e_p*v18*y_mn
 v41=-d_m*d_p*eia2*e_m*e_p*v12*y_m
 v42=d_m*d_p*eia2*e_m*e_p*v10
 v43=d_m*d_p*eia2*e_m*e_p*y_m*y_mn

 f=-v14*(((-2.d0*d_m*d_p*eia2*e_m*e_p*v17-
$  2.d0*d_m*d_p*eia2*e_m
$  *e_p)*y_mn*y_p+v24+v23+(v24+v23)*v22)*
$  y_pn+v13*((-eia2*v1*v
$  18*v2*y_mn+v19+(v16+v26)*v17)*y_p-eia2*v12*v5*v6*y_m+(v25+v
$  16)*y_m+v21)+(v19+v1*eia2*v2*v18*y_mn+(v16+v15)*v17
$  )*y_p+(v5*ei
$  a2*v6*y_m*v12+v21+(v20+v16)*y_m)*v22)
 g=v14*(((2.d0*d_p*eia2*e_p*v2-
$  2.d0*d_p*eia2*e_p*v17*v2)*y_mn*y_p+v
$  22*(-d_m*eia2*e_m*v6*y_m*y_mn-d_m*eia2*e_m*v10*v6)+d_m*
$  eia2*e_m*v6*y_m*y_mn+d_m*eia2*e_m*v6*v10)*
$  y_pn+v13*((d_m*eia2*e
$  _m*v6*v12+v29+(v31+v27)*v17)*y_p-d_m*eia2*e_m*v12*v6*y_m+(v
$  28+v30)*y_m+v32)+(-d_m*eia2*e_m*v12*v6+v29+(v28+v27)*v17)*y
$  _p+(d_m*eia2*e_m*v6*y_m*v12+v32+(v31+v30)*y_m)*v22)
 hh=-v14*(((2.d0*d_p*eia2*e_p*v17*v5-
$  2.d0*d_p*eia2*e_p*v5)*y_mn
$  *y_p-d_m*eia2*e_m*v1*y_m*y_mn+(d_m*v1*eia2*e_m*y_m*y_mn+d_m
$  *v1*eia2*e_m*v10)*v22-d_m*eia2*e_m*v1*v10)*y_pn+v13*((-
$  d_m*eia2*e_m*v1*v18*y_mn+v35+(v34+v36)*v17)*y_p+d_m*
$  v1*eia2*e_m
$  *v10*y_mn+(v37+v33)*y_m+v38)+
$  (v35+d_m*v1*eia2*e_m*v18*y_mn+(v34
$  +v33)*v17)*y_p+v22*(-d_m*eia2*e_m*v1*v10*y_mn+(v37+v36)*y_m
```

```
     $  +v38))
      i=v14*(((2.d0*d_m*d_p*eia2*e_m*e_p*v17+
     $  2.d0*d_m*d_p*eia2*e_m*e_p)*
     $  y_mn*y_p+v43+v42+(v43+v42)*v22)*y_pn+(-eia2*v12*v5*v6+v40+(
     $  v25+v39)*v17)*y_p+v22*(-eia2*v1*v10*v2*y_mn+(v39+v26)*y_m+v
     $  41)+((v5*eia2*v6*v12+v40+(v20+v39)*v17)*
     $  y_p+v41+v1*eia2*v2*v10*
     $  y_mn+(v39+v15)*y_m)*v13)

      j=0.25d0*(4.d0*m*w**2*kdiss+
     $  (1.d0-eiai)*hh*krmkp*sqrt3+(8.d0*caw-8.d0)*k
     $  p+((4.d0*eiai+4.d0)*f-8.d0)*kkr+4.d0*diss)
      k=-0.25d0*((eiai-1.0)*i*krmkp*sqrt3+(-4.d0*eiai-4.d0)*g*kkr)
      l=-0.25d0*((eiai-1.d0)*f*krmkp*sqrt3+(-4.d0*eiai-4.d0)*hh*kkp)
      mm=0.25d0*(4.0*m*w**2*kdiss+
     $  (1.d0-1.d0*eiai)*g*krmkp*sqrt3+(8.d0*caw-8.d0)*k
     $  r+((4.d0*eiai+4.d0)*i-8.d0)*kkp+4.d0*diss)
      q=-(2.d0*j*mm-2.d0*k*l)/((ii*kpi*l-ii*k*kpi)
     $  *sqrt3*saw2+(j*kpi*sqrt3**2-kpi*mm)*caw2
     $  +j*kpi*sqrt3**2+(kpi-2.d0*j)*mm+2.d0*k*l)

      uxumw=(3.d0*(1.d0+eia2)*k*kpi+
     $  sqrt3*kpi*(mm-eia2*mm))/(2.d0*eia2*(k*l-j*mm))
      uyumw=-(3.d0*(1.d0+eia2)*j*kpi+
     $  sqrt3*kpi*(l-eia2*l))/(2.d0*eia2*(k*l-j*mm))

c     system for components above crack line
      if(h.gt.1.d0) then
        w1=y_m**n
        w2=y_p**n
        w3=w1**2
        w4=w2**2
        w5=y_m**(h-0.5d0)
        w6=y_p**(h-0.5d0)
        w7=y_m**(n-h+0.5d0)
        w8=y_p**(n-h+0.5d0)
        w9=w1*w7
```

```
   w10=w2*w8
   w11=y_p**(n-h-0.5d0)
   w12=y_m**(n+h-0.5d0)
   w13=(-(d_m*e_p*(1.d0+w1)*(-1.d0+w2))+
$    d_p*e_m*(-1.d0+w1)*(1.d0+w2))*
$    (d_p*e_m*(1.d0+w1)*(-1.d0+w2)-d_m*e_p*(-1.d0+w1)*(1.d0+w2))

   uxumw=-((uxumw*v5*v6*(-1.d0+w3)*(-w10+w6)+
$    d_p*v2*(d_p*uxumw*(-1.d0+w4)*(w5-w9)+
$    e_p*uyumw*(w10+2.d0*w12*w2-(1.d0+w4)*w5+w6-w3*(w10+w6)-
$    2.d0*w2*w7+(1.d0+w4)*w9))+
$    d_m*e_m*e_p*(-(e_p*uyumw*
$    (-w10+(-1.d0+w4)*w5+w6+(-1.d0+w4)*w9+
$    (-w10+w6)*y_m**(2.d0*n)-2.d0*w1*w11*(-y_p+y_p**(2.d0*h))))+
$    d_p*uxumw*(w10-2.d0*w12*w2+(1.d0+w4)*w5+w6+w3*(w10+w6)-
$    2.d0*w2*w7+(1.d0+w4)*w9-2.d0*w1*w11*(y_p+y_p**(2.d0*h)))))/
$    (exp((ii*gn*w)/(2.d0*v))*w13))

   uyumw=-((uyumw*v1*v2*(-1.d0+w3)*(-w10+w6)+
$    e_p*v5*(e_p*uyumw*(-1.d0+w4)*(w5-w9)+
$    d_p*uxumw*(w10+2.d0*w12*w2-(1.d0+w4)*w5+w6-w3*(w10+w6)-
$    2.d0*w2*w7+(1.d0+w4)*w9))+
$    d_m*d_p*e_m*(-(d_p*uxumw*
$    (-w10+(-1.d0+w4)*w5+w6+w3*(-w10+w6)+(-1.d0+w4)*w9-
$    2.d0*w1*w11*(-y_p+y_p**(2.d0*h))))-
$    e_p*uyumw*(-w10+2.d0*w12*w2-(1.d0+w4)*w5-w6-w3*(w10+w6)+
$    2.d0*w2*w7-(1.d0+w4)*w9+2.d0*w1*w11*(y_p+y_p**(2.d0*h)))))/
$    (exp((ii*gn*w)/(2.d0*v))*w13))
 end if
```

Chapter 3

Numerical evaluation

We present the methods used to numerically evaluate the quantities presented in the previous chapter at each step in the analytical solution of the in-plane problem. We also show example results for a few systems obtained using these techniques[1].

3.1 Load calculation

The first step to numerically evaluate the components for the in-plane (Mode I) problem is calculating the load Δ as presented in Section 2.4. The load is important because it gives the boundary condition U_N that drives the crack motion. The exact relation between the two quantities in the triangular lattice is $U_N = 2u_f\Delta/\sqrt{3Q_0}$. The Δ formula is reproduced here

$$\Delta = \exp\left[-\int \frac{d\omega}{4\pi}\left(\frac{\omega_0^2}{i\omega'(\omega_0^2+\omega'^2)}[\ln Q(\omega') - \overline{\ln Q(\omega')}] + \frac{\omega_0 \ln|Q(\omega')|^2}{\omega_0^2+\omega'^2}\right)\right]. \tag{3.1}$$

This calculation has been done before by Marder[22]. First we make the change of variables $\omega = s/\sqrt{1-s^2}$. This makes the bounds of the integral [0,1].

[1] This chapter uses material from the paper [3] by Chris Behn and Michael Marder, titled 'The transition from subsonic to supersonic cracks' and published in the journal *Phil. Trans. R. Soc. A* in 2015. Michael Marder edited the text.

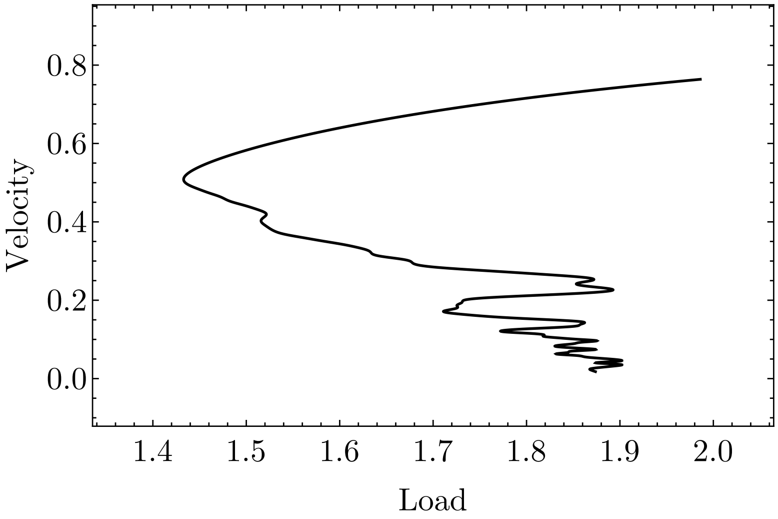


Figure 3.1: Scaled crack velocity v/c versus load Δ for $N = 10$, $k_{\|} = 1$, $k_{\perp} = 0$. Stokes and Kelvin dissipation are $b = 0.1$ and $\beta = 0.01$.

Romberg integration is then used to evaluate this integral numerically. The iterative process is stopped when the Δ error is less than 10^{-5}. Figure 3.1 shows Δ calculated for a range of crack velocities v in this manner.

The integration is performed in a Fortran script, part of which can be seen at the end of the full component calculation script in Appendix A.

3.2 Fourier transform approximation

There are many Fourier transforms that need to be evaluated to compute the components. Evaluating these analytically is impossible, as the algebraic expressions are too large (see Section 2.5). We are forced to use numerical approximations to do the Fourier transforms. Most continuous Fourier transforms (CFT) can be approximated by a discrete Fourier transform (DFT). The steps to do so can be found in [2]. We give a quick overview of them here.

First, the CFT of a complex function $f(t)$ and its inverse are:

$$F[f](\omega) = \int_{-\infty}^{\infty} dt\, f(t) e^{-it\omega} \tag{3.2a}$$

$$F^{-1}[f](t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} d\omega\, f(t) e^{it\omega}. \tag{3.2b}$$

The DFT of an n-long sequence z and its inverse are:

$$D_k(z) = \sum_{j=0}^{n-1} z_j e^{-2\pi ijk/m} \tag{3.3a}$$

$$D_k^{-1}(z) = \frac{1}{m} \sum_{j=0}^{n-1} z_j e^{2\pi ijk/m}. \tag{3.3b}$$

DFTs can be evaluated quickly by using a variant of the Fast Fourier Transform (FFT) algorithm. We use the freely available FFTW[2] algorithm, well known for its speed and accuracy. Its Fortran implementation is easily added to all our Fortran scripts.

CFTs are approximated using DFTs as follows. Assume the function $f(t)$ that is being Fourier transformed is zero outside the interval $(-L/2, L/2)$. We want to sample $f(t)$ m times, with m being a power of 2 (a requirement for using the FFTs). The time difference between sampled points in the interval is $\beta = L/m$. The time values are given by $t_j = (j - m/2)\beta$, with $0 \leq j < m$. The frequency values in the output are $\omega_k = 2\pi(k - m/2)/L = 2\pi(k - m/2)/(m\beta)$,

[2] Fastest Fourier Transform in the West, (www.fftw.org)

with $0 \leq k < m$. Now write the CFT as

$$\begin{aligned}
F(\omega_k) &= \int_{-\infty}^{\infty} dt\, f(t) e^{-it\omega_k} \\
&= \int_{-L/2}^{L/2} dt\, f(t) e^{-it\omega_k} \\
&\approx \sum_{j=0}^{m-1} f(t_j) e^{-it_j\omega_k} \beta \\
&= \beta \sum_{j=0}^{m-1} f(t_j) e^{-2\pi i(j-m/2)(k-m/2)/m} \\
&= \beta e^{\pi i(k-m/2)} \sum_{j=0}^{m-1} f(t_j) e^{\pi ij} e^{-2\pi ijk/m} \\
&= (-1)^k \beta D_k[(-1)^j f(t_j)].
\end{aligned} \tag{3.4}$$

This form can be rapidly evaluated using an FFT algorithm.

3.3 Component calculation

All atomic positions in time can be evaluated from rapid numerical operations involving nothing but algebra and Fourier transforms. First we sample the function $Q(\omega)$ $n = 2^p$ times over an interval of width $2\omega_{\max}$ and use Fourier transforms to compute $U^-(\omega)$ and $\mathbf{u}_n(\omega)$ from Eq. (2.18). Before taking the inverse Fourier transform, any singular behavior of the form $1/(i\omega)$ must be subtracted from $u_n^y(\omega)$. The coefficient is determined from the asymptotic behavior of $u_n^y(t)$ and is given by

$$\lim_{\omega\to 0}[(i\omega)u_n^y(\omega)] = (Q_0 - U_N)\left(1 - \frac{(n-1/2)}{N}\right). \tag{3.5}$$

The subtracted function is then added back analytically after the inverse Fourier transform. This completes the numerical calculation of $\mathbf{u}_n(t)$. We can check the accuracy of the components knowing the exact asymptotic behavior along the crack line:

$$\lim_{t\to\pm\infty} u^x_{1/2}(t) = 0, \quad \lim_{t\to-\infty} u^y_{1/2}(t) = Q_0, \quad \lim_{t\to+\infty} u^y_{1/2}(t) = U_N. \tag{3.6}$$

Also, in the $\beta \to 0$ limit, the load is given by $\Delta = Q^-(0)/\sqrt{Q_0}$. This can be compared to direct numerical integration of Eq. (2.37). We can choose a cutoff frequency $\omega_{\max}$ that minimizes the difference between these two Δ values. Figure 3.2 shows an example of this minimization. For a given system, we can find the value of $\omega_{\max}$ that minimizes the error in Δ by running our script at a low sampling resolution $n = 2^p$, then using it to find $\omega_{\max}$ for higher resolutions. The cutoff frequency increases by a factor of $\sqrt{2}$ each time the resolution increases by a factor of 2. For higher resolutions than those shown in Figure 3.2, the error in Δ becomes very small. At $n = 2^{19}$, the error is about 1%, and up to $n = 2^{23}$, the error decreases to roughly 0.5%. The time cutoff $t_{\max}$ and interval Δt between output points are given in terms of $\omega_{\max}$ by

$$t_{\max} = \frac{n\pi}{2\omega_{\max}} \tag{3.7a}$$

$$\Delta t = \pi/\omega_{\max}. \tag{3.7b}$$

Figure 3.3 shows the horizontal and vertical components of atomic position calculated in this fashion. Figure 3.4 shows the vertical components for three different rows calculated using the equations of motion above the crack

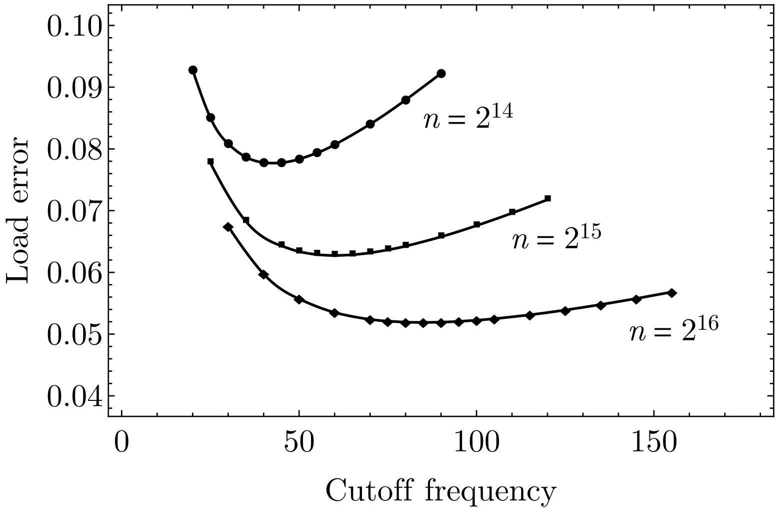


Figure 3.2: Difference in load Δ values obtained from direct integration and analytical component solutions in a system with 10 rows. As the resolution n=2^p increases by a factor of 2, the cutoff frequency $\omega_{\max}$ increases by a factor of $\sqrt{2}$. At $n = 2^{19}$, the error is about 1%, and up to $n = 2^{23}$, the error decreases to roughly 0.5%.

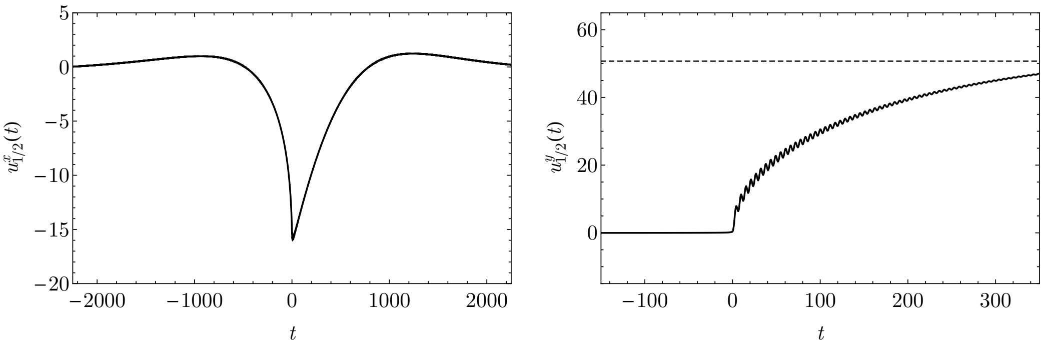
line written down in Section 2.5. We choose the boundary condition U_N in such a fashion that at the moment bonds break they have increased in length by $u_f = 1$. Thus we say that all displacements in the plots are measured in units of u_f. Keeping the breaking length fixed means in turn that all the computations correspond to physical systems with the same fracture energy. As the number of rows $2(N+1)$ increases, the extension U_N needed to bring the system to the point of fracture increases as $\sqrt{N}$. For supersonic cracks, the extensions become large multiples of u_f.

It is worth pausing to ask how Figure 3.3 might be produced were one not using the Wiener-Hopf technique. It would be possible. It would require integrating the equations of motion for a crack in a system 1002 rows high.

In order for atomic motions to reach steady state, the crack would need to run for a distance around ten times the height of the system [15]. Although this could be sped up with a cutting and pasting procedure, accurate results would require a system 3000 columns long. Thus one would have to run to steady state a system with around 3 million atoms; this would require a supercomputer. By contrast, producing Figure 3.3 required just a bit over a second using a single processor on a laptop.

Originally, a Mathematica[3] script was used to numerically evaluate the lengthy algebraic expressions in Section 2.5, and perform the Fourier transforms. We found that the former section of the code took up over 90% of the total computing time of the script. It was decided that we would convert the Mathematica script to Fortran, given its ability to do numerical calculations quickly. For example, the Mathematica script took over 15 minutes to evaluate the solutions in Figure 3.3, while the Fortran script takes roughly 2 seconds for the same calculation on the same computer, a 2012 MacBook Pro. Converting the script to Fortran provided the necessary speed boost to finish the calculations presented in the next chapters.

[3] Wolfram, (www.wolfram.com/mathematica)



(a) Horizontal

(b) Vertical

Figure 3.3: Horizontal and vertical components for the displacement of an atom versus time. The atom sits on the crack line in a system with 1002 rows ($N = 500$), crack speed $v/c = 0.7$, where c is the Rayleigh wave speed, $k_{\|} = 1$, $k_{\perp} = 0$, and Stokes and Kelvin dissipation of $b = 10^{-4}$, $\beta = 10^{-2}$. The dashed line indicates the asymptotic vertical displacement of the atom, U_N. All displacements are measured in units of u_f, the extension at which bonds break. The atom is nearly motionless until the crack arrives at $t = 0$. The horizontal component oscillates slightly, then returns to zero. The vertical component approaches the boundary condition U_N, with high frequency, small amplitude oscillations lasting for a time on the order of $1/\beta$. The high frequency oscillations are phonons that result from the periodic snapping of atomic bonds.

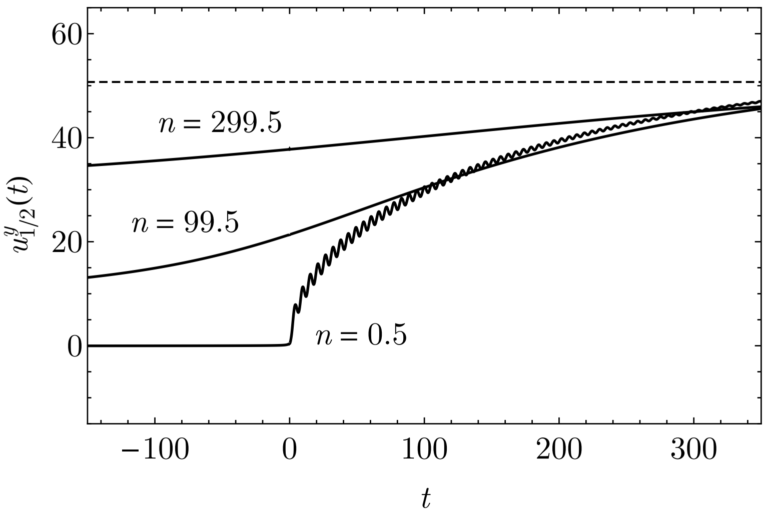


Figure 3.4: Vertical components for the displacements of three different rows of atoms versus time. Shown are rows 1, 100 and 300 above the crack line in a system with 1002 total rows ($N = 500$). The crack speed is $v/c = 0.7$, $k_{\|} = 1$, $k_{\perp} = 0$, and Stokes and Kelvin dissipation are $b = 10^{-4}$, $\beta = 10^{-2}$. The dashed line indicates the asymptotic vertical displacement of the atom, U_N. Note how higher rows tend toward this vertical boundary condition for all time. Row 500 above the crack line has this displacement for all t.

Chapter 4

Behavior of the lattice solutions

We examine the behavior of in-plane solutions for a range of subsonic speeds and for a few low supersonic speeds. We find that subsonic solutions exhibit familiar high frequency, small amplitude oscillations. The shape of the vertical displacement near the crack tip is in agreement with continuum results for subsonic speeds. We find that supersonic solutions exhibit new low frequency, large amplitude oscillations. This behavior appears in subsonic speeds close to the Rayleigh wave speed. Finally, we present the results of an experiment that validate some of these calculations[1].

4.1 Supersonic transition

Having developed a tool that allows us very quickly to find the time history of atoms in the vicinity of a running crack, we now turn to the question with which we began. We ask what happens near a crack tip as the motion of the crack moves from subsonic to supersonic motion. The important wave speed is the Rayleigh wave speed c. For central forces ($k_\perp = 0$), it is given

[1] This chapter uses material from the paper [3] by Chris Behn and Michael Marder, titled 'The transition from subsonic to supersonic cracks' and published in the journal *Phil. Trans. R. Soc. A* in 2015. Michael Marder edited the text.

explicitly in our model by

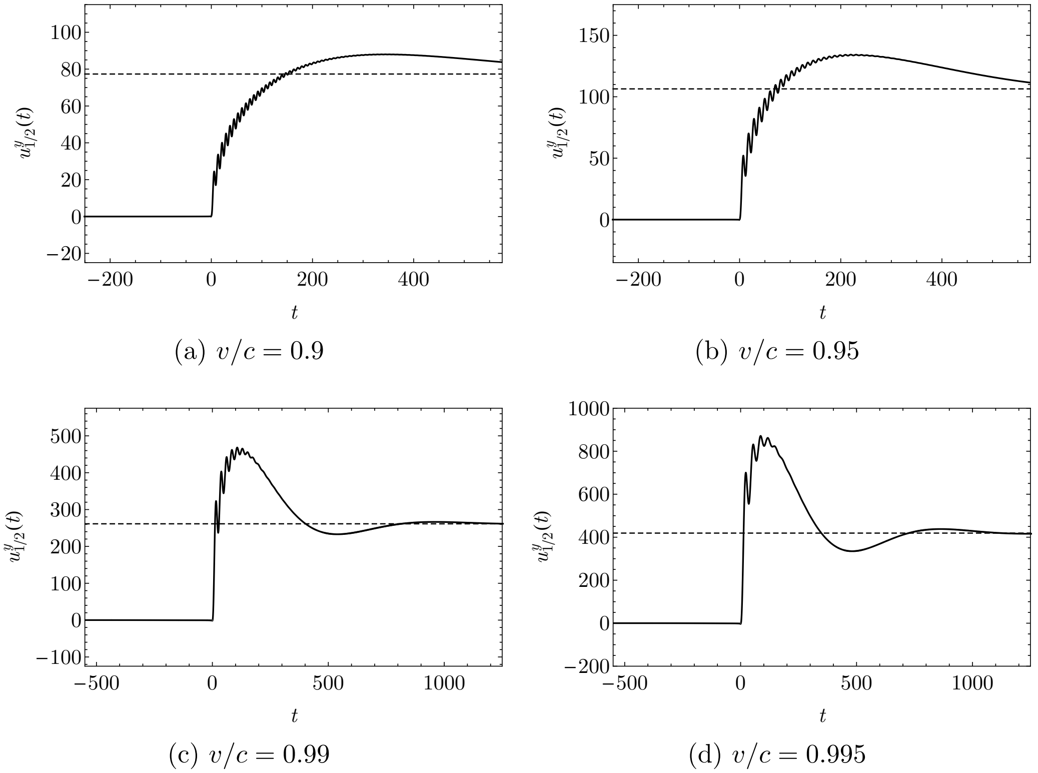
$$c = \sqrt{2(1 - \sqrt{3}/3)}c_t, \tag{4.1}$$

which is the root of the denominator in Eq. (1.35), and the limiting speed for cracks in continuum fracture mechanics.

Supersonic solutions in the lattice model look very different from subsonic solutions. Figures 4.1 and 4.2 show how the vertical displacement $u^y_{1/2}(t)$ of an atom on the crack line varies as the crack speed increases through the Rayleigh wave speed from $v = 0.9c$ to $v = 1.05c$. We can see two distinct behaviors for subsonic and supersonic cracks. In both cases, the atom is nearly motionless until the crack approaches. After the crack passes, in subsonic solutions, the vertical displacement approaches the boundary condition U_N with small amplitude, high frequency oscillations that continue for a time on the order of $1/\beta$ as shown in Figure 4.1a. These are phonons carrying energy left over after the bonds along the crack line have snapped [9].

In supersonic solutions, the vertical displacement also approaches the boundary displacement, but with large amplitude, low frequency oscillations as seen in Figure 4.2d. The phonons of the subsonic solutions are no longer present in supersonic solutions. Interestingly, the subsonic phonons and supersonic large oscillations appear simultaneously in the subsonic solutions just below the Rayleigh wave speed, as we see in Figures 4.1c-d and Figure 4.2a.

The continuum solutions for in-plane fracture are graphed in [4]. Using



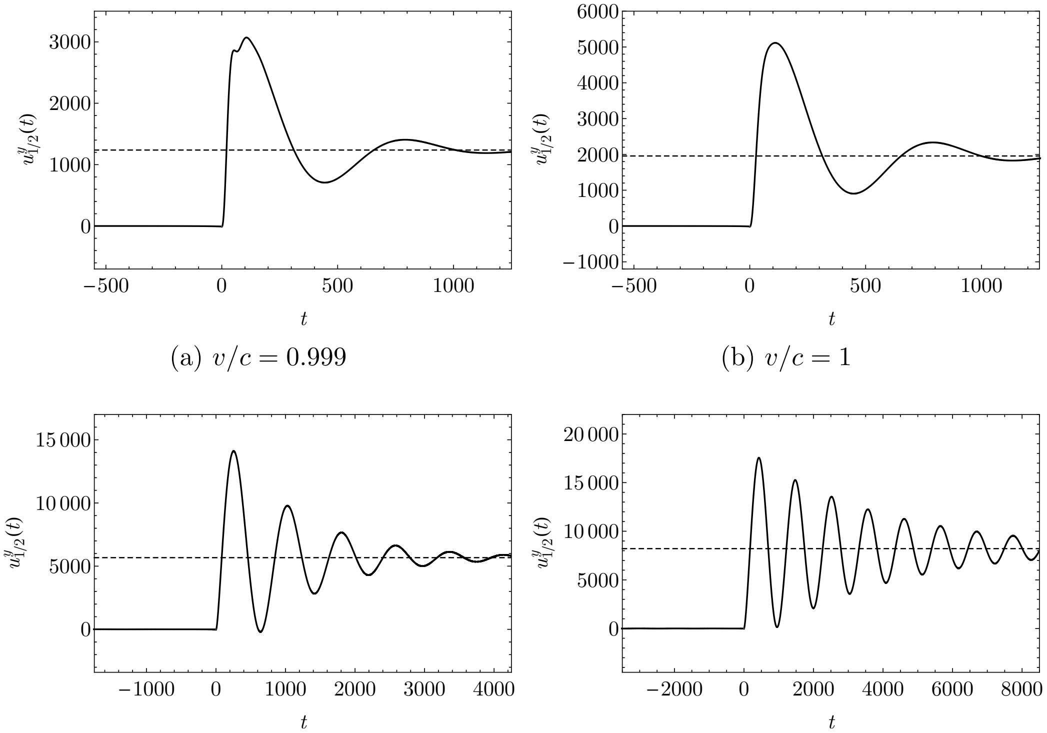
(a) $v/c = 0.9$

(b) $v/c = 0.95$

(c) $v/c = 0.99$

(d) $v/c = 0.995$

Figure 4.1: Vertical component $u^y_{1/2}(t)$ along the crack line in a system with 1002 rows ($N = 500$) as the crack speed v increases through the Rayleigh wave speed c. Displacements are measured in units of u_f, the extension for which bonds break. Note the small amplitude, high frequency oscillations for subsonic solutions (a), compared to the large amplitude, low frequency oscillations for supersonic solutions. Subsonic solutions close to c exhibit both these behaviors (d).For all computations, $k_\parallel = 1$, $k_\perp = 0$, Stokes and Kelvin dissipation are $b = 10^{-4}$ and $\beta = 10^{-2}$. The dashed line indicates the asymptotic vertical displacement of the atom, U_N.



(a) $v/c = 0.999$

(b) $v/c = 1$

(c) $v/c = 1.01$

(d) $v/c = 1.05$

Figure 4.2: Vertical component $u^y_{1/2}(t)$ along the crack line in a system with 1002 rows ($N = 500$) as the crack speed v increases through the Rayleigh wave speed c. Displacements are measured in units of u_f, the extension for which bonds break. Note the large amplitude, low frequency oscillations for supersonic solutions (d), compared to the small amplitude, high frequency oscillations for subsonic solutions. In panel (c) the atom passes below its original height after time $t = 0$. Thus this computation is only consistent for a model where a bond that breaks never reforms. For all computations, $k_\parallel = 1$, $k_\perp = 0$, Stokes and Kelvin dissipation are $b = 10^{-4}$ and $\beta = 10^{-2}$. The dashed line indicates the asymptotic vertical displacement of the atom, U_N.

the same notation as Eq. (1.35), the vertical displacement is:

$$u^y = \frac{2\alpha}{\sqrt{2\pi}\mu R}[2 - (1 + \beta^2)]\sqrt{vt}K_I \tag{4.2}$$

where $R = 4\alpha\beta - (1 + \beta^2)^2$, and $x = vt$ is the horizontal position of the crack tip as it moves in steady state. This same $\sqrt{t}$ behavior of u^y is present in the subsonic lattice solutions shown in Figures 3.3b and 4.1a. We could use this square root displacement to extract a stress intensity factor K_I from the discrete system, as a complement to methods involving energy balance used in the past. However, as the crack speed increases toward the Rayleigh wave speed, the square root profile becomes increasingly indistinct, and it is is completely lost at the Rayleigh speed and above.

As an illustration of the difference between subsonic and supersonic solutions, in Figure 4.3 we plot the asymptotic behavior of subsonic and supersonic solutions for atomic motion near the crack tip. For the subsonic solutions, the displacement rises as $\sqrt{t}$ as expected. For supersonic solutions, it rises instead as $\sqrt{t}^3$.

4.2 Varying the system size

The subsonic lattice solutions have a vertical displacement $u^y_{1/2}(t)$ that matches the continuum result Eq. (4.2), superposed with small amplitude, high frequency oscillations that result from periodic bond breaking, and carry off all the energy flux to the crack tip not absorbed by the bond breaking process.

The supersonic solutions feature large amplitude, low frequency oscil-

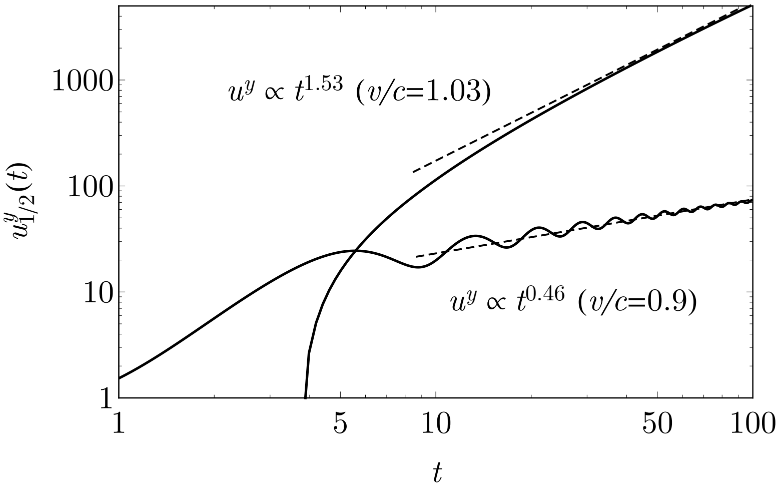


Figure 4.3: Comparison of subsonic and supersonic solutions for atoms near the crack tip. Both systems are of height $2(N+1)$ with $N = 800$.

lations. There is a simple explanation for these oscillations, which take place at what we call the supersonic block frequency ω_{b}. This frequency is what we find if the lattice oscillates vertically as a block, with the top row held fixed. We can solve for this frequency using the equations of motion along the crack line Eqs. (2.10) and (2.11). These oscillations occur after the crack has passed, so we can eliminate the $\theta(t)$ functions multiplying $U^{-}(t)$. Fourier transforming these gives Eqs. (2.13) and (2.14) with $U^{-}(\omega)$ replaced by zero. The components $\mathbf{u}_{3/2}(\omega)$ can be written as linear combinations of the $\mathbf{u}_{1/2}(\omega)$ components, which gives us a system to solve. The determinant of this system for $\mathbf{u}_{1/2}(\omega)$ must vanish, which lets us solve for ω_{b}, the lowest normal mode frequency. The coefficients $C_n^k(\omega)$ in Eqs. (2.22) and (2.23) are so lengthy that this procedure is extremely cumbersome and not very informative.

It is more useful to consider the lattice as a continuous block and use

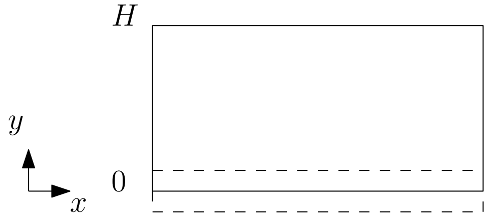


Figure 4.4: Diagram of the isotropic elastic block. It is held fixed at $y = H$ and is free to oscillate at $y = 0$.

the equations of linear elasticity [10]. The top of the block is fixed while the bottom is free to oscillate, as shown in Figure 4.4. If the relaxed height of the block is H, the lowest normal mode frequency can be shown to be

$$\omega_{\text{b}} = (\pi/H)c_l. \tag{4.3}$$

We can derive Eq. (4.3) as follows. Consider an isotropic elastic block held fixed at $y = H$ and free to oscillate at $y = 0$. The boundary conditions are:

$$\mathbf{u}(x, H) = \mathbf{0} \tag{4.4}$$

$$\sigma_{xy}(x, 0) = \sigma_{yy}(x, 0) = 0. \tag{4.5}$$

The linear elasticity equations are:

$$\rho\frac{\partial^2\mathbf{u}}{\partial t^2} = \mu\nabla^2\mathbf{u} + (\lambda + \mu)\nabla(\nabla \cdot \mathbf{u}) \tag{4.6}$$

$$\sigma_{ij} = \lambda\delta_{ij}u_{k,k} + \mu(u_{i,j} + u_{j,i}). \tag{4.7}$$

The displacement $\mathbf{u}$ can be decomposed into longitudinal and transverse components $\mathbf{u} = \mathbf{u}^L + \mathbf{u}^T$. The components are derived from the potentials

$\phi^L = -iAe^{i(kx+k_L y-\omega t)}$ and $\phi^T = -iBe^{i(kx+k_T y-\omega t)}$ such that:

$$\mathbf{u}_L = \nabla\phi^L = Ae^{i(kx+k_L y-\omega t)}(k\hat{\mathbf{x}} + k_L\hat{\mathbf{y}}) \tag{4.8}$$

$$\mathbf{u}_T = -\phi^T_y\hat{\mathbf{x}} + \phi^T_x\hat{\mathbf{y}} = Be^{i(kx+k_T y-\omega t)}(-k_T\hat{\mathbf{x}} + k\hat{\mathbf{y}}). \tag{4.9}$$

Written like this, we can set them to be real. The wave numbers are also real since we are looking for oscillatory solutions.

The full expression for the displacement is

$$\mathbf{u} = e^{i(kx-\omega t)}[(Ake^{ik_L y} - Bk_T e^{ik_T y})\hat{\mathbf{x}} + (Ak_L e^{ik_L y} + Bke^{ik_T y})\hat{\mathbf{y}}]. \tag{4.10}$$

Boundary condition (4.4) gives:

$$Ak\cos(k_L H) - Bk_T\cos(k_T H) = 0 \tag{4.11}$$

$$Ak_L\cos(k_L H) + Bk\cos(k_T H) = 0. \tag{4.12}$$

The determinantal equation of this system is

$$(k^2 + k_L k_T)\cos(k_L H)\cos(k_T H) = 0. \tag{4.13}$$

We are looking for solutions with nonzero A and B, therefore we must have $\cos(k_L H) = 0$ and $\cos(k_T H) = 0$.

The Navier-Cauchy equation (4.6) gives two possible frequencies:

$$\omega_T^2 = (k^2 + k_T^2)c_T^2 \tag{4.14}$$

$$\omega_L^2 = (k^2 + k_L^2)c_L^2 \tag{4.15}$$

with $c_T = \sqrt{\mu/\rho}$ and $c_L = \sqrt{(\lambda+2\mu)/\rho}$. Similarly, the second boundary condition (4.5) gives

$$\frac{k^2 - k_T^2}{2kk_L} = \frac{2\mu k k_T}{\lambda(k^2+k_L^2)+2\mu k_L^2}. \tag{4.16}$$

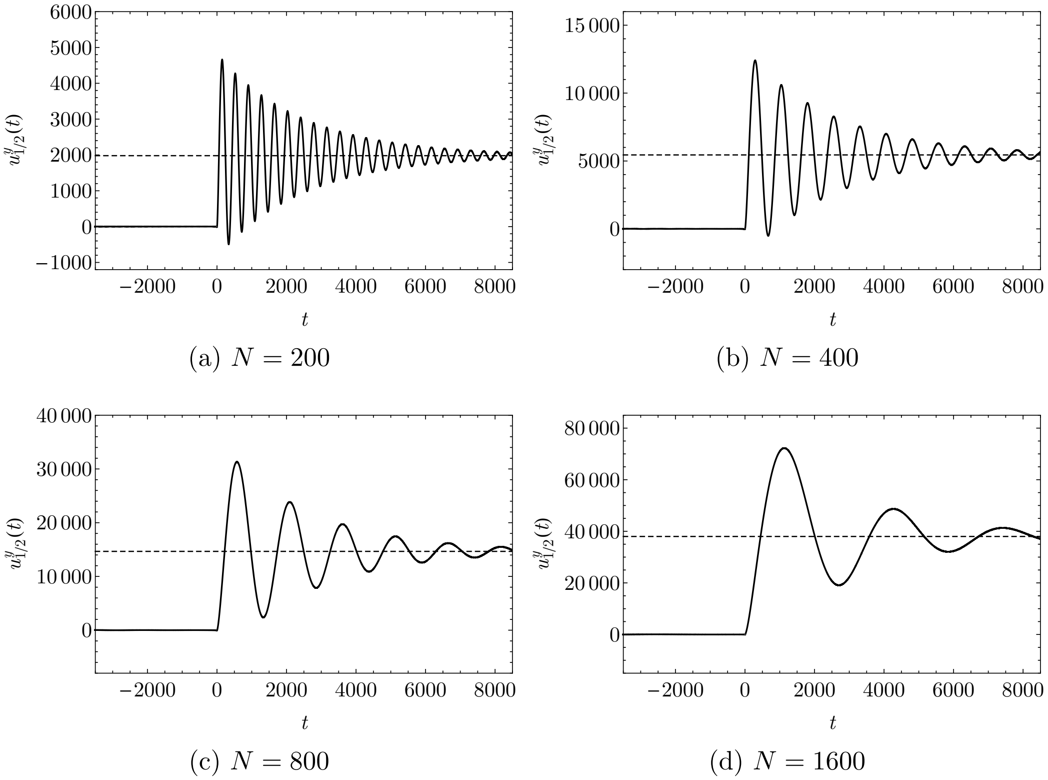
The solution that best matches the supersonic block frequency obtained from the numerical calculations in the triangular lattice is ω_L in Eq. (4.15) and $k_T = k_L = \pi/(2H)$ in Eq. (4.13). Eq. (4.16) then gives

$$k^2 = \frac{\lambda+2\mu}{\lambda}k_L^2 = \frac{k_L^2}{1-2c_T^2/c_L^2}. \tag{4.17}$$

In the triangular lattice $c_T/c_L = 1/\sqrt{3}$, so $k^2 = 3k_L^2$. Also $k_L = \pi/(2H)$, $H = N\sqrt{3}/2$ and $c_L = 3\sqrt{K/8}$ (K is the spring constant) so the supersonic block frequency is

$$\omega_{\mathrm{b}} = 2k_L c_L = \frac{2\sqrt{3}\pi}{N}\sqrt{\frac{K}{8}}. \tag{4.18}$$

This expression is approximate to the extent that our lattice system is not actually a continuum, and because the waves are not infinite in horizontal wavelength. Figure 4.5 shows how the vertical displacement along the crack line for a fixed supersonic speed varies with the system size. As the system grows from 402 rows high to 3202 rows high, the oscillation frequency decreases in proportion. A fit to the calculated results gives $\omega_{\mathrm{b}} \sim 3.1742/N$. This result agrees with the result in Eq. (4.3) within 5%. Figure 4.6 shows the vertical displacements for three different rows above the crack line at the supersonic speed $v/c = 1.01$. Note that they all share the same low frequency, high



(a) $N = 200$

(b) $N = 400$

(c) $N = 800$

(d) $N = 1600$

Figure 4.5: Vertical component $u^y_{1/2}(t)$ along the crack line for supersonic crack speed $v/c = 1.03$ as the number of atomic rows $2(N+1)$ increases. Displacements are measured in units of u_f, the extension for which bonds break. Note that the large amplitude oscillation frequency depends inversely on the system size as $\omega_{\mathrm{b}} \sim 3.1742/N$. In panels (a) and (b), the oscillations take atoms below their original height. Thus these computations are only correct for a model in which bonds that once break never reform. For all computations, $k_\| = 1$, $k_\perp = 0$, Stokes and Kelvin dissipation are $b = 10^{-4}$ and $\beta = 10^{-2}$. The dashed line indicates the asymptotic vertical displacement of the atom, U_N.

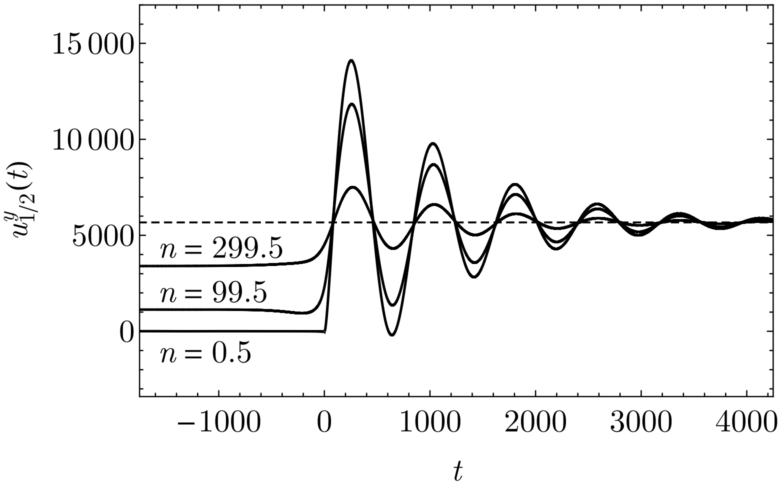


Figure 4.6: Vertical components for the displacements of three different rows of atoms versus time. Shown are rows 1, 100 and 300 above the crack line in a system with 1002 total rows ($N = 500$). The crack speed is $v/c = 1.01$, $k_\parallel = 1$, $k_\perp = 0$, and Stokes and Kelvin dissipation are $b = 10^{-4}$, $\beta = 10^{-2}$. The dashed line indicates the asymptotic vertical displacement of the atom, U_N. Note how higher rows tend toward this vertical boundary condition for all time. Row 500 above the crack line has this displacement for all t. The low frequency, high amplitude oscillations are consistent with the lattice oscillating as a continuous block after the crack has passed.

amplitude oscillations. This is consistent with the idea of the lattice oscillating as a continuous block after the crack has passed.

We have seen how the vertical displacement along the crack line changes as the crack speed becomes supersonic for a fixed system size (Figures 4.1 and 4.2). The subsonic solutions just below the Rayleigh wave speed have both small amplitude, high frequency oscillations as well as large amplitude, low frequency oscillations. The low frequency oscillations as seen in Figure 4.2d begin to appear in Figures 4.1c and 4.1d alongside the high frequency ones, which disappear completely at the Rayleigh speed (Figure 4.2b) and above.

In Figure 4.7 we examine the vertical motion of atoms along the crack line for a fixed subsonic speed as the system size varies. For the small system with 202 rows ($N = 100$) the solution shares the character of both subsonic and supersonic solutions. Phonon oscillations are visible, but at the same time there is a long-wavelength oscillation with the frequency of the supersonic block frequency ω_{b}. Now we increase the system size and monitor the location of the first long-wavelength peak. As one can see in Figure 4.7 it slides to the right and diminishes in amplitude. The location of the peak is given approximately by $t_{\text{b}} \sim 0.7150N$, just as the period of the large amplitude oscillations goes as $T \propto N$ in Figure 4.5. The height of the peak decreases slower than $1/\sqrt{N}$ and approaches the boundary condition U_N asymptotically from above as seen in Figure 4.7d. Thus for any given subsonic crack speed below Rayleigh wave speed, there exists a system size sufficiently large to produce behavior like the continuum solution Eq. (4.2), and make it possible to find a stress intensity factor.

4.3 Convergence for large systems

Considering again the subsonic solutions for large N, it is important to note that there is a convergence near the origin, or close to the crack tip. This can be seen as an overlap in Figure 4.8 near $t = 0$ of solutions for the systems $N = 800$ and $N = 1600$.

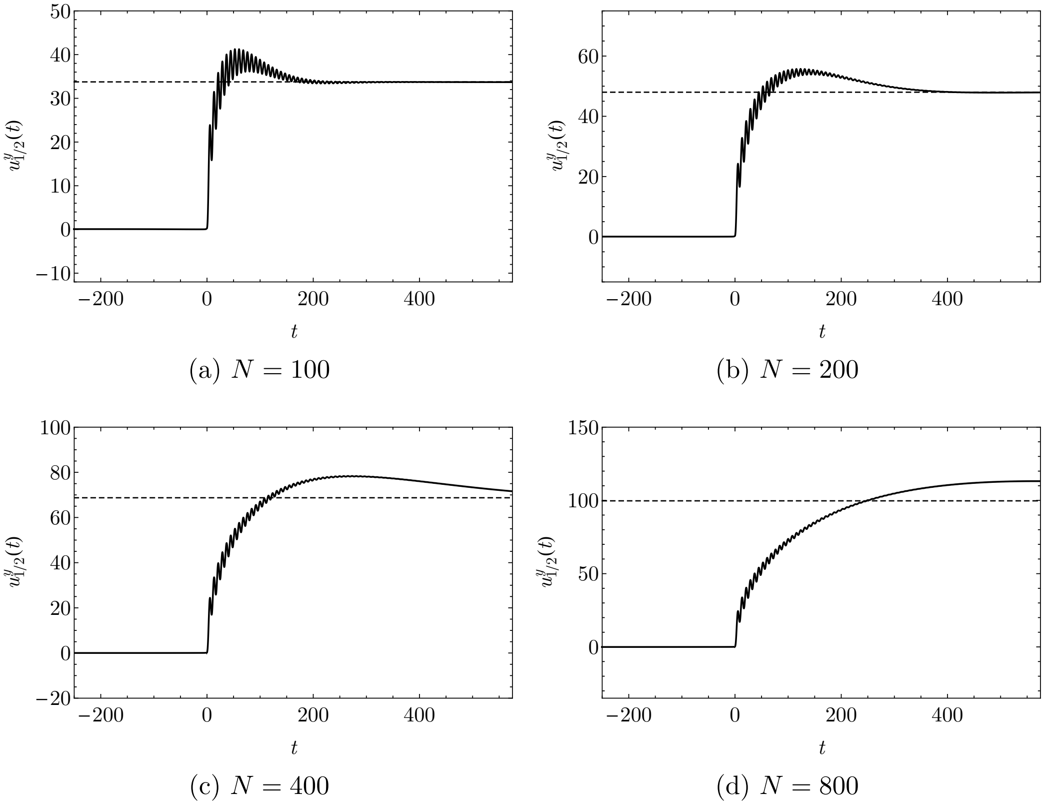
We run into convergence issues with the computation of the load Δ using the Romberg technique mentioned in Section 3.1 for systems with $N > 3000$. However, this convergence near the origin suggests that very large N solutions can be approximated by smaller N solutions.

4.4 Fineberg experiment

Boué, Livne and Fineberg conducted an experiment[2] to measure the shape change of the supersonic crack tip opening displacement in an effectively infinite strip. These measurements in the strip geometry showed a "tadpole" like shape around the crack tip as seen in Figure 4.9. The near tip form of the crack tip has a parabolic shape due to a stress singularity, and far from the tip the crack displacement transitions to a constant Δ, but the displacement first overshoots Δ before converging far from the crack tip.

Both of these features are present in our analytical solutions. The

[2] J. Fineberg, private communication (2015)



(a) $N = 100$

(b) $N = 200$

(c) $N = 400$

(d) $N = 800$

Figure 4.7: Vertical component $u^y_{1/2}(t)$ along the crack line for subsonic crack speed $v/c = 0.9$ as the number of horizontal rows of atoms $2(N+1)$ increases. Displacements are measured in units of u_f, the extension for which bonds break. Note how the small peak seen in (a) diminishes and approaches the boundary condition U_N asymptotically from above for larger N, and its position in time varies as $t \sim 0.7150N$. For all computations, $k_\parallel = 1$, $k_\perp = 0$, Stokes and Kelvin dissipation are $b = 10^{-4}$ and $\beta = 10^{-2}$. The dashed line indicates the asymptotic vertical displacement of the atom, U_N.

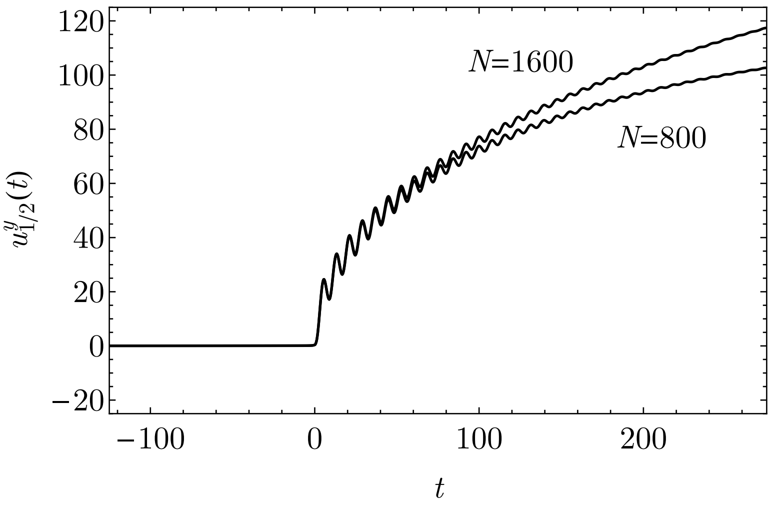


Figure 4.8: Convergence near the crack tip for large lattices. Each system has a height $2(N+1)$.

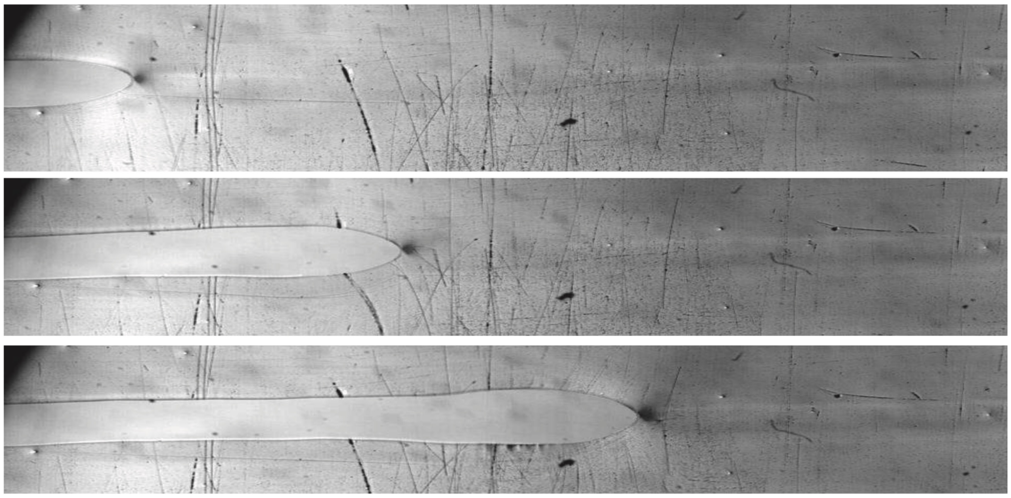


Figure 4.9: Experiment by Boué, Livne and Fineberg showing "tadpole" form around the crack tip in an effectively infinite strip.

overshoot or “tadpole” is present in the solutions in Figure 4.7. The overshoot is a feature of the long wavelength oscillations characteristic of the supersonic solutions. Its height diminishes as the system size N increases. After the overshoot, the vertical displacement approaches the boundary condition U_N asymptotically from above, as seen in the experimental snapshots in Figure 4.9.

Chapter 5

Energy flux in the lattice

Continuum fracture mechanics tells us that the energy flow into the crack tip becomes negative or imaginary for supersonic cracks, which would allow us to conclude that supersonic cracks are unphysical. This result is shown in Eq. (1.35). Supersonic cracks do in fact exist experimentally and in the lattice solutions presented in Chapter 4. Therefore, motivated by the continuum energy result, we study energy flow in the lattice. First, we construct an energy flux vector for the triangular lattice inspired by the Poynting vector in electromagnetic theory, then we make vector plots at each lattice point for the subsonic and supersonic solutions calculated in the previous chapter.

5.1 Poynting vector

We want to construct an energy flux vector for our lattice to look at the energy flow. Before doing so, we can think about the Poynting vector from electromagnetism. Here is a quick derivation of it [12]. We start with the total energy stored in electromagnetic fields in a volume V

$$U = \frac{1}{2}\int_V \left(\epsilon E^2 + \frac{1}{\mu}B^2\right) dV. \tag{5.1}$$

This can be derived by considering energy conservation in electrodynamics. Consider a charge and current configuration producing fields $\mathbf{E}$ and $\mathbf{B}$ at a time t. We want to find the work dW done by the electromagnetic forces on a charge q in a time interval dt. The Lorentz force law says

$$dW = \mathbf{F} \cdot d\mathbf{L} = q(\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \mathbf{v} dt = q\mathbf{E} \cdot \mathbf{v} dt. \tag{5.2}$$

Let $q = \rho dV$ and $\mathbf{J} = \rho \mathbf{v}$. The time derivative of the work done on all charges in a volume V is

$$\frac{dW}{dt} = \int_V (\mathbf{E} \cdot \mathbf{J}) dV. \tag{5.3}$$

The quantity $\mathbf{E} \cdot \mathbf{J}$ is the power per unit volume. We can eliminate $\mathbf{J}$ using the Maxwell-Ampère law ($\mu\mathbf{J} + \mu\epsilon\partial\mathbf{E}/\partial t = \nabla \times \mathbf{B}$) to get

$$\mathbf{E} \cdot \mathbf{J} = \frac{1}{\mu}\mathbf{E} \cdot (\nabla \times \mathbf{B}) - \epsilon\mathbf{E} \cdot \frac{\partial \mathbf{E}}{\partial t}. \tag{5.4}$$

We use the vector relation

$$\nabla \cdot (\mathbf{E} \times \mathbf{B}) = \mathbf{B} \cdot (\nabla \times \mathbf{E}) - \mathbf{E} \cdot (\nabla \times \mathbf{B}) \tag{5.5}$$

and Faraday's law ($\partial\mathbf{B}/\partial t = -\nabla \times \mathbf{E}$) to get

$$\mathbf{E} \cdot (\nabla \times \mathbf{B}) = -\mathbf{B} \cdot \frac{\partial \mathbf{B}}{\partial t} - \nabla \cdot (\mathbf{E} \times \mathbf{B}). \tag{5.6}$$

Using

$$\mathbf{E} \cdot \frac{\partial \mathbf{E}}{\partial t} = \frac{1}{2}\frac{\partial}{\partial t}(E^2), \quad \mathbf{B} \cdot \frac{\partial \mathbf{B}}{\partial t} = \frac{1}{2}\frac{\partial}{\partial t}(B^2) \tag{5.7}$$

we get

$$\mathbf{E} \cdot \mathbf{J} = -\frac{1}{2}\frac{\partial}{\partial t}\left(\epsilon E^2 + \frac{1}{\mu}B^2\right) - \frac{1}{\mu}\nabla \cdot (\mathbf{E} \times \mathbf{B}). \tag{5.8}$$

Finally, plugging this into Eq. (5.3) and using the divergence theorem for the second term gives

$$\frac{dW}{dt} = -\frac{d}{dt}\int_V \frac{1}{2}\left(\epsilon E^2 + \frac{1}{\mu}B^2\right) dV - \int_A \frac{1}{\mu}(\mathbf{E}\times\mathbf{B})\cdot d\mathbf{A} \tag{5.9}$$

where A is the surface boundary of the volume V. Equation (5.9) is a statement of Poynting's theorem, which says that the work done by the electromagnetic forces on charges in a volume is equal to the change in energy stored in the electromagnetic fields, minus the energy flowing out of the volume's surface. The vector inside the area integral is the Poynting vector:

$$\mathbf{S} \equiv \frac{1}{\mu}(\mathbf{E}\times\mathbf{B}). \tag{5.10}$$

It is an energy flux density vector, with units of energy per time, per area. The product $\mathbf{S}\cdot d\mathbf{A}$ is the energy flowing through the surface $d\mathbf{A}$ per unit time.

5.2 Lattice energy flux vector

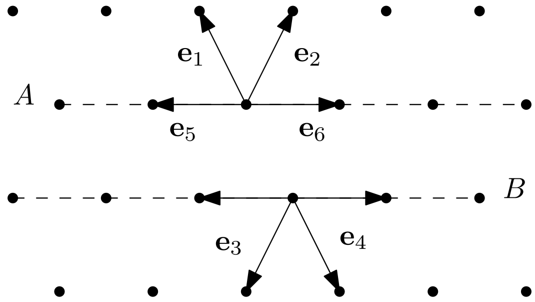


Figure 5.1: Diagram of the triangular lattice. Lines A and B are rows above and below the crack line. The $\mathbf{e}_j$ are the six different displacement vectors along the lattice: $\mathbf{e}_1 = (-1/2, \sqrt{3}/2)$, $\mathbf{e}_2 = (1/2, \sqrt{3}/2)$, $\mathbf{e}_3 = (-1/2, -\sqrt{3}/2)$, $\mathbf{e}_4 = (1/2, -\sqrt{3}/2)$, $\mathbf{e}_5 = (-1, 0)$, $\mathbf{e}_6 = (1, 0)$.

We now want to construct an energy flux vector in the lattice. First, consider the lattice geometry shown in Figure 5.1. Lines A and B are rows along the crack line. The $\mathbf{e}_j$ are the six displacement vectors for the triangular lattice. Atoms along rows A and B are eventually separated by the traveling crack.

Let us work with an atom on line A. The rate at which work is done, or power, by the atom in the direction $\mathbf{e}_1$ is given by $\mathbf{F}_1 \cdot \mathbf{v}_A$, where $\mathbf{v}_A$ is the velocity of the atom on A. The result for the atom in the direction $\mathbf{e}_2$ is $\mathbf{F}_2 \cdot \mathbf{v}_A$. The power in the vertical direction is the sum of these two values, and should equal the y-component of the energy flux vector $\mathbf{P}$ in the lattice

$$P_{y,A} = (\mathbf{F}_1 + \mathbf{F}_2) \cdot \mathbf{v}_A. \tag{5.11}$$

This represents the energy flow through the atom across the line A. Similarly, we can show for an atom on line B that

$$P_{y,B} = -(\mathbf{F}_3 + \mathbf{F}_4) \cdot \mathbf{v}_B. \tag{5.12}$$

The average of these two values can be taken as the y-component of a general energy flux vector in the lattice. Another way to get this result is by defining the energy flux vector at each lattice site to be

$$\mathbf{P} = \sum_{j=1}^{6} (\mathbf{F}_j \cdot \mathbf{v})\mathbf{e}_j \tag{5.13}$$

where $\mathbf{v}$ is the velocity of the atom at the lattice site. The y-component of this vector is the same, with a $\sqrt{3}$ factor. We can think of this formula as the sum of the six directional vectors $\mathbf{e}_j$, each weighted by the power $(\mathbf{F}_j \cdot \mathbf{v})$ along their respective direction. We will use the energy flux vector $\mathbf{P}$ defined in Eq. (5.13) in the rest of this work.

The forces $\mathbf{F}_j$ and the full expressions for the components of the energy flux vector $\mathbf{P}$ are given in this portion of the Fortran script shown below for the atoms above the crack line. The full script is shown in Appendix B.

```
c       displacements along all six directions
        d1x=above(c+(col-gn)*nt,1)-above(c,1)
        d1y=above(c+(col-gn)*nt,2)-above(c,2)
        d2x=above(c+(col-gn+1)*nt,1)-above(c,1)
        d2y=above(c+(col-gn+1)*nt,2)-above(c,2)
        d3x=above(c-(col+gn)*nt,1)-above(c,1)
        d3y=above(c-(col+gn)*nt,2)-above(c,2)
        d4x=above(c-(col+gn-1)*nt,1)-above(c,1)
        d4y=above(c-(col+gn-1)*nt,2)-above(c,2)
        d5x=above(c-1*nt,1)-above(c,1)
        d5y=above(c-1*nt,2)-above(c,2)
        d6x=above(c+1*nt,1)-above(c,1)
        d6y=above(c+1*nt,2)-above(c,2)
c       displacement dot products dj.ej
c       force along each direction
        d1e1=d1x*e1x+d1y*e1y
        f1x=kp*e1x*d1e1
        f1y=kp*e1y*d1e1
        d2e2=d2x*e2x+d2y*e2y
        f2x=kp*e2x*d2e2
        f2y=kp*e2y*d2e2
        d3e3=d3x*e3x+d3y*e3y
        f3x=kp*e3x*d3e3
        f3y=kp*e3y*d3e3
        d4e4=d4x*e4x+d4y*e4y
        f4x=kp*e4x*d4e4
        f4y=kp*e4y*d4e4
        d5e5=d5x*e5x+d5y*e5y
        f5x=kp*e5x*d5e5
        f5y=kp*e5y*d5e5
        d6e6=d6x*e6x+d6y*e6y
        f6x=kp*e6x*d6e6
        f6y=kp*e6y*d6e6
c       velocity components
        vx=above(c,3)
        vy=above(c,4)
c       dot products fj.v
c       energy flux components in each direction
        f1v=f1x*vx+f1y*vy
        p1x=e1x*f1v
        p1y=e1y*f1v
        f2v=f2x*vx+f2y*vy
        p2x=e2x*f2v
        p2y=e2y*f2v
        f3v=f3x*vx+f3y*vy
        p3x=e3x*f3v
        p3y=e3y*f3v
        f4v=f4x*vx+f4y*vy
        p4x=e4x*f4v
        p4y=e4y*f4v
        f5v=f5x*vx+f5y*vy
        p5x=e5x*f5v
        p5y=e5y*f5v
        f6v=f6x*vx+f6y*vy
        p6x=e6x*f6v
        p6y=e6y*f6v
c       total energy flux
        px=p1x+p2x+p3x+p4x+p5x+p6x
        py=p1y+p2y+p3y+p4y+p5y+p6y
```

5.3 Vector plots

Given the lattice energy flux vector, we can now make vector plots of the energy flow in the lattice around the the crack tip. The procedure to generate such plots is as follows. First, we compute the displacements and velocities of all atoms in a region around the crack tip. We use the analytical procedure presented in Chapter 2 to find the solution for each row, and the results are computed using a Fortran script similar to the one presented in Appendix A. The position and velocity components are then fed into another script that computes the energy flux vector **P** at each atom, averages it over a period $T = a/v$, and outputs its components and the positions of each atom so we can make a vector plot. The full energy flux Fortran script is shown in Appendix B. The Mathematica script used to make the vector plots is shown in Appendix C.

For every vector plot, the position of each atom is indicated by a black dot, with the corresponding energy flux vector **P** centered on it. The vector magnitude is indicated by color, not length on the plot. We use the "Rainbow" color scheme from Mathematica, which goes (purple → blue → green → yellow → orange → red) as the vector magnitude varies along $(0, 1)$. The vector magnitudes are scaled first logarithmically, then linearly along $(0, 1)$ with 0 and 1 being the minimum and maximum magnitudes in the lattice region, respectively.

Figure 5.2 shows a vector plot generated in such a way. The system size is $N = 500$ and the crack speed is subsonic with $v/c = 0.6$. The crack tip is shown moving to the right in the positive x-direction. We set the bond breaking length to be $u_f = 1$ like all the plots in Chapter 4. This leads to large values of the boundary condition U_N, which makes the lattice difficult to visualize after the crack has passed. For all other vector plots, we will set the boundary condition to be $U_N = 1$.

Figures 5.3-5.10 show vector plots for various crack speeds above and below the Rayleigh wave speed c for a system with $N = 500$ and $U_N = 1$. It is useful to look at the corresponding vertical component plots in Figures 3.3b, 4.1 and 4.2 when examining these vector plots. We know from our discussions in Chapter 4 that subsonic solutions feature high frequency, small amplitude oscillations, while supersonic solutions feature low frequency, large amplitude oscillations. The first peak of the supersonic oscillations is present in subsonic solutions close to the Rayleigh wave speed. Figures 5.3-5.8 show energy flows around the crack opening to the left of tip that drive the small amplitude oscillations in the subsonic solutions. The energy flow is strongest right at the crack tip where the crack is opening just after $t = 0$. Figures 5.9 and 5.10 show similar energy flows on a different time scale, driving the large amplitude oscillations in the supersonic solutions.

We can approximate the total energy flux out of the crack in the lattice in a way analogous to the result Eq. (1.35) from continuum fracture mechanics. The total energy flow out of a row above the crack line is

$$\sum_c \mathbf{P} \cdot \hat{\mathbf{n}} = \sum_c P_y \tag{5.14}$$

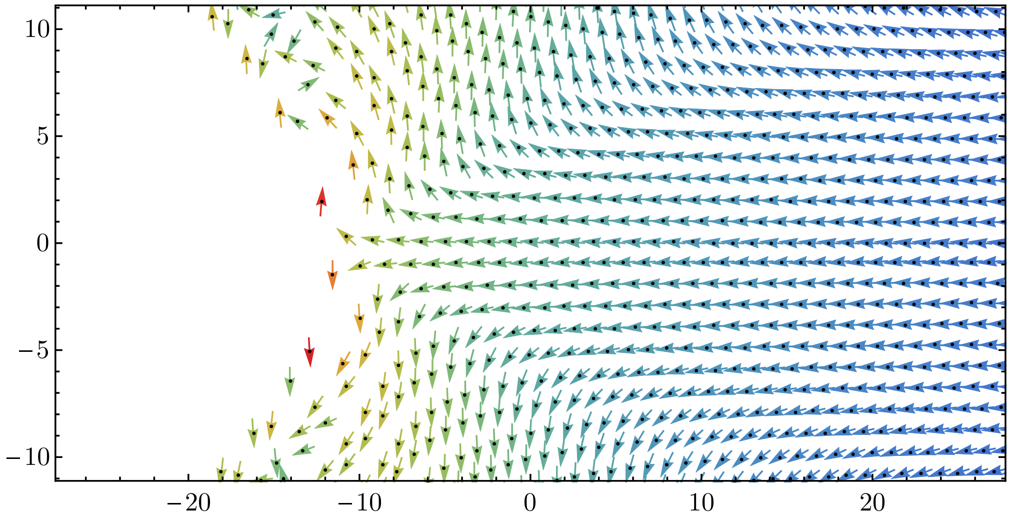


Figure 5.2: Vector plot for $v/c = 0.6$ with $N = 500$ and $u_f = 1$.

where the index c runs over a number of columns on both sides of the crack tip, and $\hat{\mathbf{n}}$ is an outward unit vector normal to the row. Above the crack line $\hat{\mathbf{n}} = \hat{\mathbf{y}}$, and below the crack line $\hat{\mathbf{n}} = -\hat{\mathbf{y}}$. Figure 5.11 shows the total flux out of 200 columns as a function of the row n. The result is a positive outward flow for both subsonic and supersonic solutions.

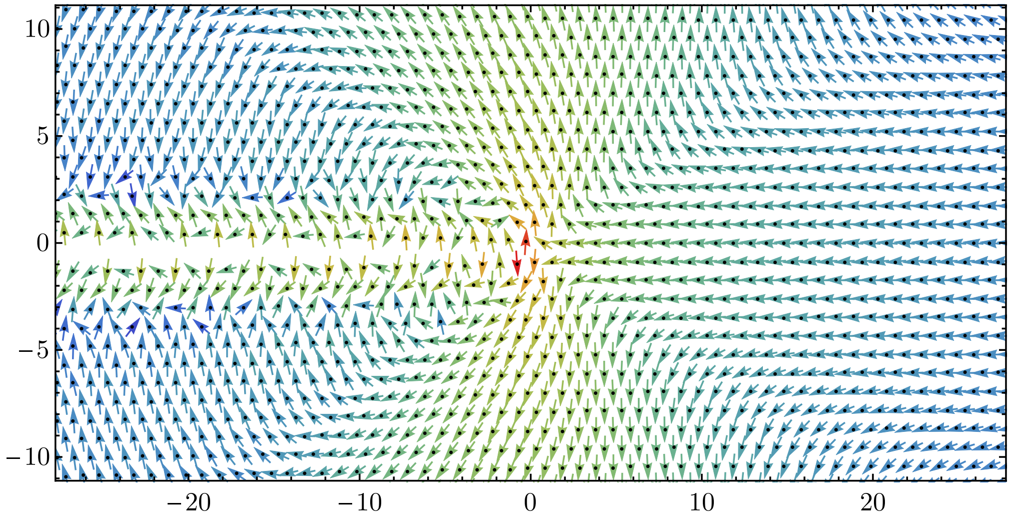


Figure 5.3: Vector plot for $v/c = 0.7$ with $N = 500$ and $U_N = 1$.

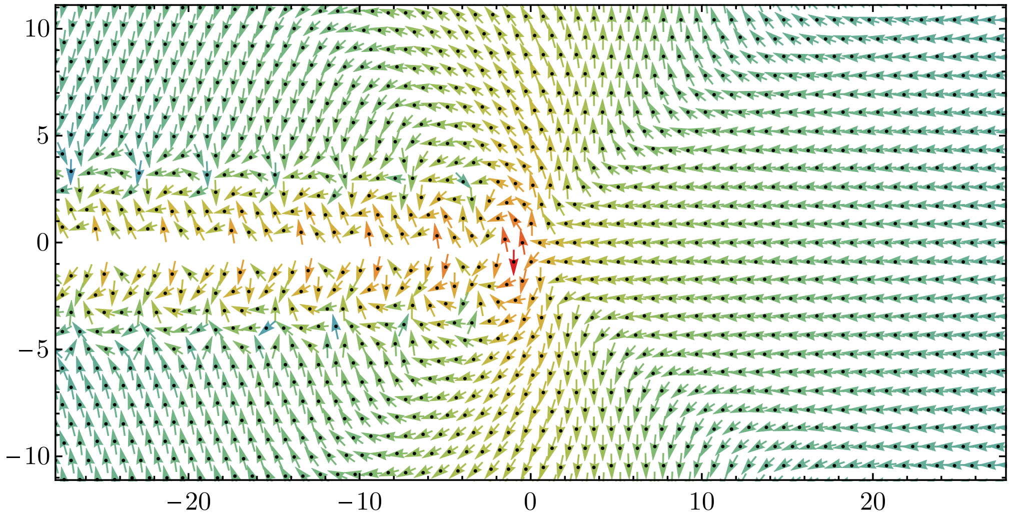


Figure 5.4: Vector plot for $v/c = 0.9$ with $N = 500$ and $U_N = 1$.

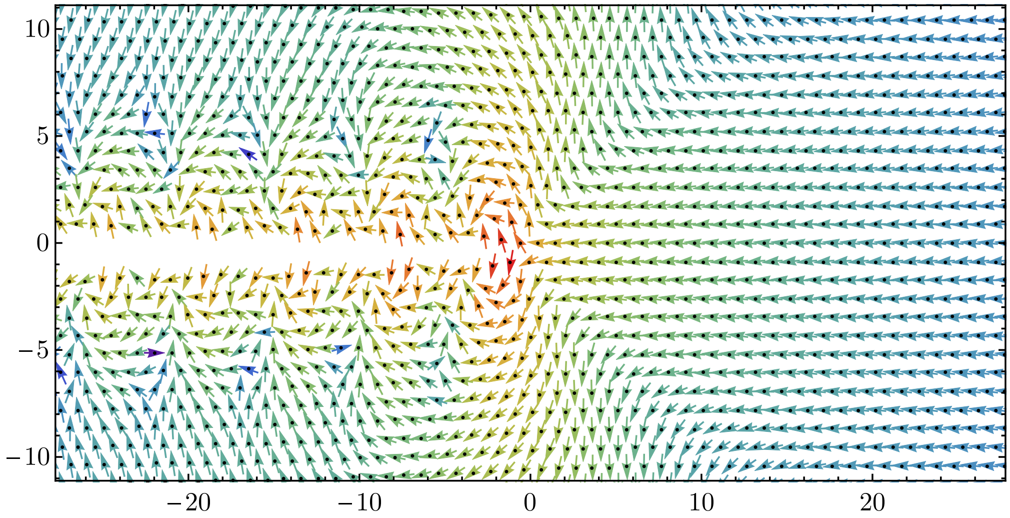


Figure 5.5: Vector plot for $v/c = 0.95$ with $N = 500$ and $U_N = 1$.

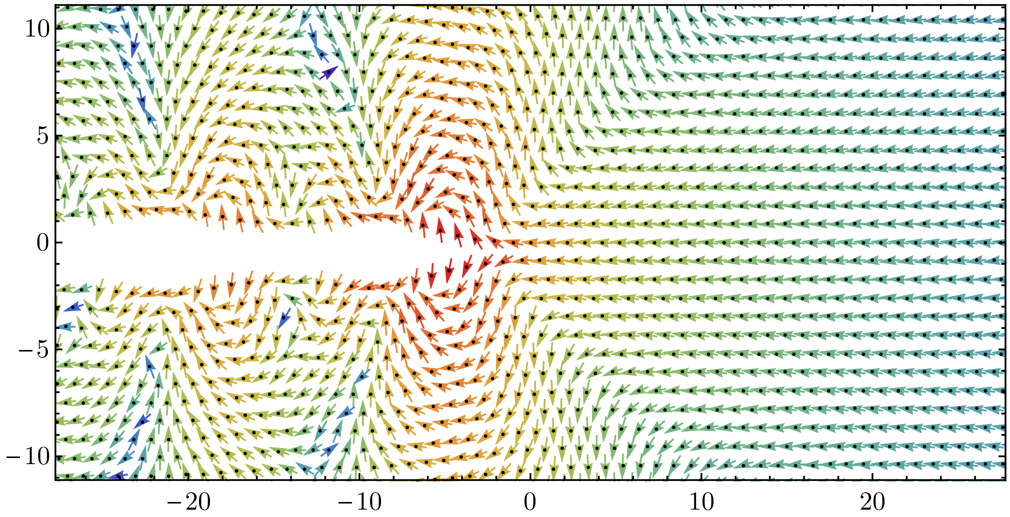


Figure 5.6: Vector plot for $v/c = 0.99$ with $N = 500$ and $U_N = 1$.

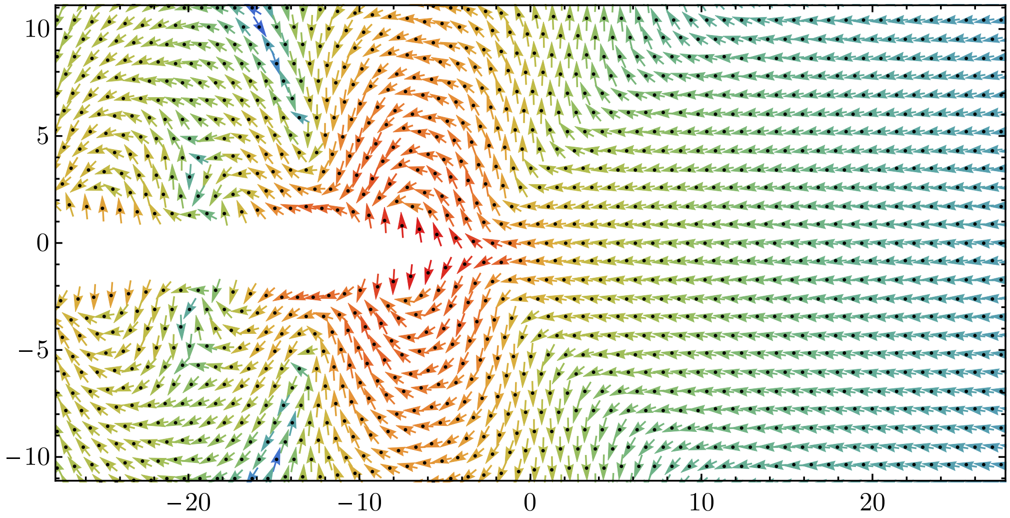


Figure 5.7: Vector plot for $v/c = 0.995$ with $N = 500$ and $U_N = 1$.

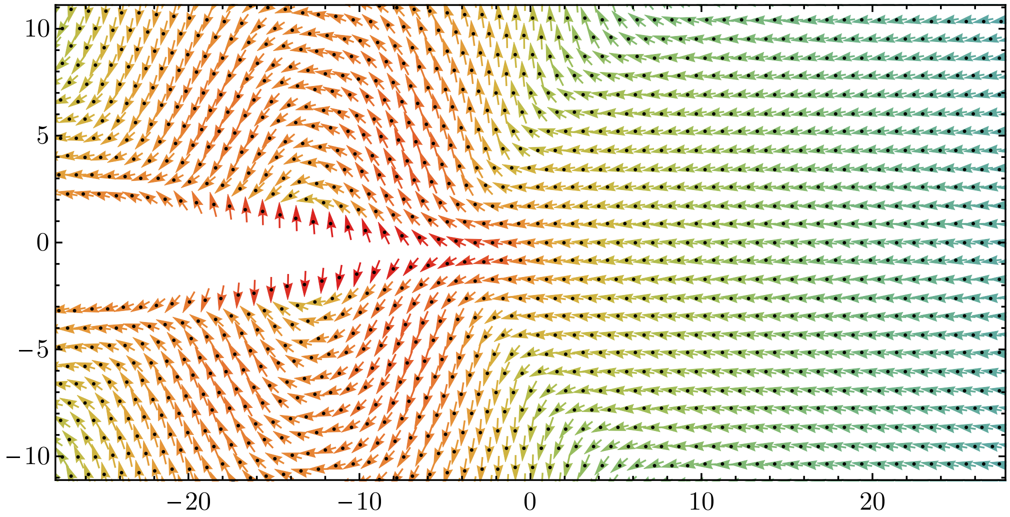


Figure 5.8: Vector plot for $v/c = 0.999$ with $N = 500$ and $U_N = 1$.

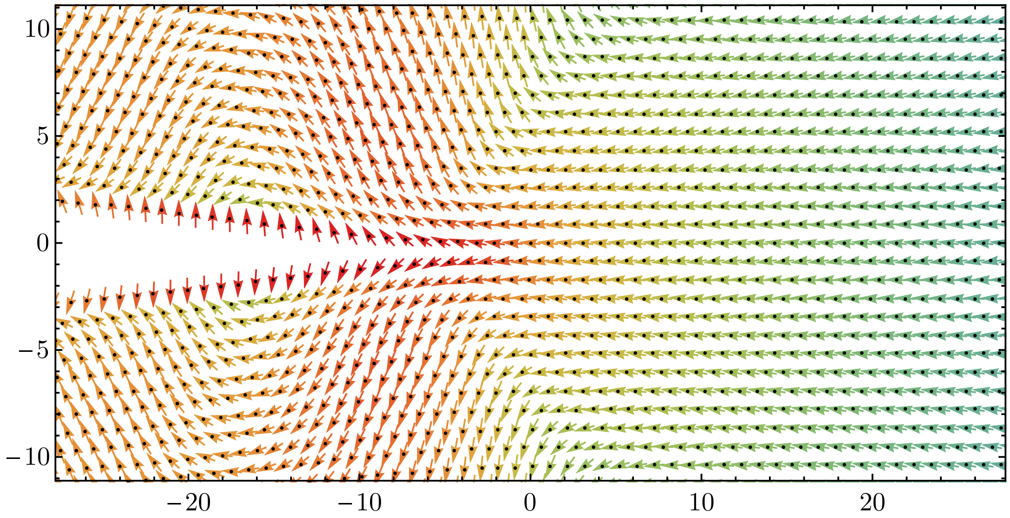


Figure 5.9: Vector plot for $v/c = 1$ with $N = 500$ and $U_N = 1$.

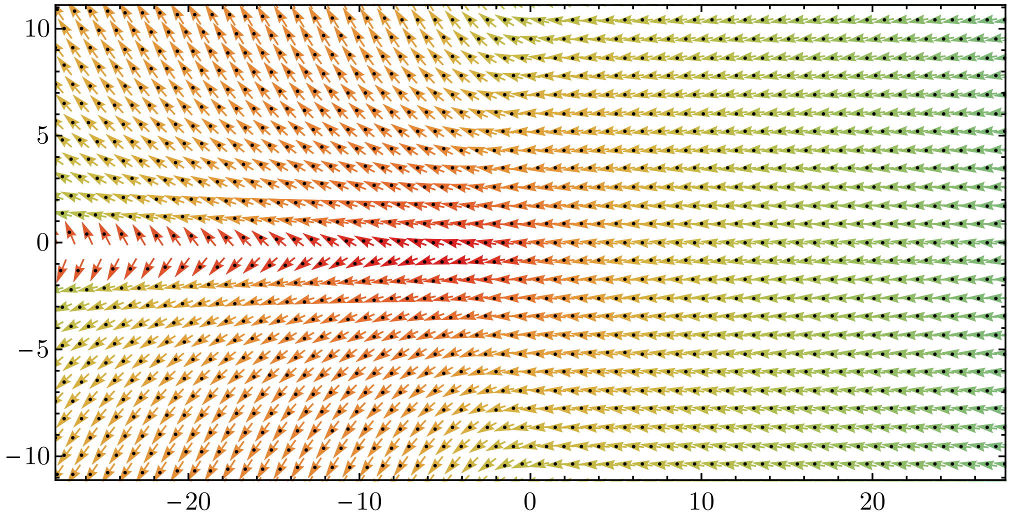
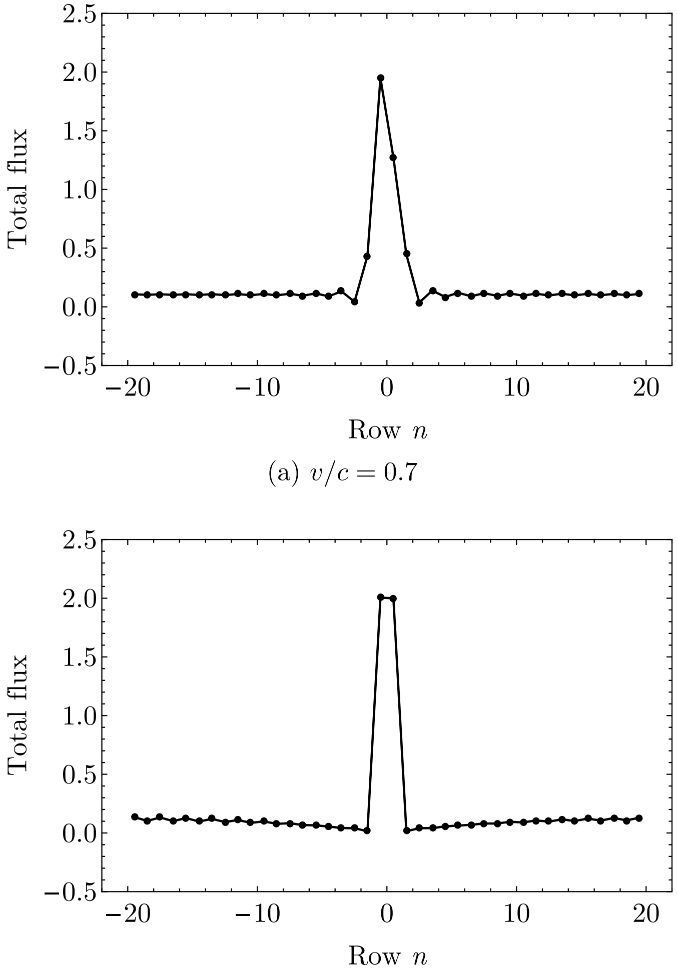


Figure 5.10: Vector plot for $v/c = 1.01$ with $N = 500$ and $U_N = 1$.



(a) $v/c = 0.7$

(b) $v/c = 1.01$

Figure 5.11: Total energy flux in the y-direction out of 200 columns of atoms around the crack tip for a number of rows.

Chapter 6

Conclusion

We have presented the full analytical solution for atomic motions accompanying steady state in-plane crack motion in a brittle triangular lattice. This solution allows rapid numerical evaluation of atomic motion for very large systems, which is necessary to make comparisons with results from continuum fracture mechanics.

Eq. (1.35) has been interpreted to mean that cracks cannot propagate faster than the Rayleigh wave speed. However supersonic solutions do exist in lattice systems. Subsonic solutions are characterized by small amplitude, high frequency oscillations in the time dependence of the vertical displacement of an atom along the crack line. The overall envelope of the displacement agrees with the continuum result in Eq. (4.2). Supersonic solutions are characterized by large amplitude, low frequency oscillations, which can be attributed to the lattice oscillating as a block after the crack has passed. We call this frequency the supersonic block frequency ω_{b}, and show that its value can be approximated using linear elasticity. We have also shown that subsonic solutions close to the Rayleigh wave speed exhibit small amplitude, high frequency and large amplitude, low frequency oscillations at the same time. This mixing of the character of subsonic and supersonic solutions can be eliminated by increasing the system size N sufficiently.

There are three natural objections to what we have presented. The first is that systems 2000 atoms high are computationally challenging, but they are a far cry from macroscopic systems 10^{12} atoms high. Thus we have not really reached the macroscopic limit. The second is that we are drawing general conclusions about crack behavior, but we have solved only a particular atomic model with very special and particularly simple interatomic forces. The third is that since the experiments of Schardin [29] it has been known that cracks do not reach the Rayleigh wave speed, let alone exceed it, so discussions of supersonic cracks are not relevant.

In response to the first objection, we have focused on scaling our solutions in such a fashion that the large N behavior becomes apparent. For example, in Figure 4.7 we assert that in a system with $N = 10^{12}$ the plot of vertical displacement versus time for an atom just above the crack tip would look indistinguishable from 4.7d. This can be seen in Figure 4.8. In Figure 4.5, we assert that in a system with $N = 10^{12}$, the period of the large oscillations would continue to increase in proportion to N, and in order for the oscillations to remain visible the dissipation would need to decrease accordingly.

In response to the second objection, we invoke the universality of fracture mechanics. Although our atomic force laws are much simpler than any forces that genuinely would arise from the quantum mechanics of interactions, they capture the essence of brittle solids, and include all of dynamic linear elastic fracture mechanics as a special case in the long-wavelength limit. Thus we believe that conclusions we draw about the continuum (large N) limit of our theory should have general validity. The details of what happens on small scales should be indicative of what would happen in real brittle systems, but would ultimately depend upon realistic microscopic details. For examples of what more realistic systems might look like, see for example [15].

In response to the third objection, we note that cracks in brittle isotropic materials are usually limited by dynamic instabilities to terminal velocities below the Rayleigh wave speed [9]. However, in some cases the instabilities are suppressed and supersonic cracks become possible. Rubber provides one instance [25], and materials with highly anisotropic fracture energy might provide others. Supersonic cracks are not inevitable just because instabilities have been suppressed [17]: sufficient energy must be supplied to drive cracks above the Rayleigh wave speed. But they do become possible.

The calculations presented here are just the first application of this method. There are some additional questions still pending. Because the stress intensity factor stops being defined as the crack passes the Rayleigh wave speed, the conventional continuum view of energy flux stops applying. This observation provides an unsatisfying answer to the question of why the crack speed is not limited by energy flux. Once the crack travels faster than the Rayleigh wave speed, it is clear that it must extract the energy needed for bond breaking from a local environment of finite size.

We have begun to answer some of these questions by looking at the energy flow in our lattice solutions. We constructed an energy flux vector to make plots of the energy flow over a region including the crack tip. The vector plots show energy flows around the crack tip that drive oscillations in both the subsonic and supersonic solutions. An estimate of the total energy flux out of a rows around the crack line shows that the energy flowing into the crack tip is positive for both subsonic an supersonic crack speeds.

Appendices

Appendix A

Component script

This is the Fortran script we wrote to compute the components of the in-plane solution solved analytically in Chapter 2, and numerically evaluated in Chapters 3 and 4. The input parameters are the system size N, the dissipations b and β, spring constants $k_{\|}$, $k_{\perp}$, and $k_{\|}^{I}$, the crack speed ratio v/c and the row number. The sample resolution $n = 2^p$ sufficient to make accurate plots increases with the crack speed. For subsonic plots with $v/c < 1$, $p = 19$. For supersonic plots with v/c between 1 and 1.05, p ranges from 20 to 23.

```
      program mode1
      implicit complex*16 (a-h,o-z)
      common/a/n,m,a,b,bk,kp,kr,kpi,v
      external qcomp
      include "fftw3.f"

      real*8 n,m,a,b,bk,kp,kr,kpi,v
      real*8 uf,Un,Q0,h,vr,w,kkr,kkp,krmkp
      complex*16 ii,i,j,k,l,mm,kdiss
      real*8 smin,smid,smax,ans1,ans2,ans,delta,dtest

c     number of sample points np=2^p2
c     number of plotted points nplot
      integer p,p2,np,nplot
      parameter(p2=19,np=2**p2,nplot=2**17)
      real*8 wmax,tmax,dt,dw,t0
      integer gn,n0

c     fftw arrays/plans
      complex*16 lnQ(np),uxum(np),uyum(np),lnQt(np),Qm(np)
      complex*16 Um(np),ux(np),uy(np),uxt(np),uyt(np)
      integer*8 planLNQT,planQM,planUXT,planUYT

c     lattice parameters
      n=500.d0
      m=1.d0
      a=1.d0
      b=0.0001d0
      bk=0.01d0
      kp=1.d0
      kr=0.d0
      kpi=1.d0
      Q0=1.d0/(1.d0+2.d0*n)

c     rayleigh velocity
      vr=sqrt((2.d0/3.d0)*(3.d0-sqrt(3.d0)))*
     $  a*sqrt(3.d0*kp/(8.d0*m))

c     crack velocity in terms of rayleigh velocity;
c     h-th row height=h+0.5, h=0 on crack line
      v=0.7d0*vr
      h=0.5d0

c     triangular lattice factor
      gn=mod(int(h-0.5d0),2)

c     constants
      pi=3.1415926535897931
      ii=cmplx(0.d0,1.d0)
      sqrt3=sqrt(3.d0)

c     fourier transform parameters
```

```
      wmax=60.d0*sqrt(2.d0)**(p2-15)
      tmax=pi*np/(2.d0*wmax)
      dt=(2.d0*tmax)/np
      dw=(2.d0*pi)/(np*dt)

c     load integration
      smin=0.d0
      smid=0.2d0
      smax=0.95d0
      call qromb(qcomp,smin,smid,ans1)
      call qromb(qcomp,smid,smax,ans2)
      ans=ans1+ans2
      delta=exp(ans*2.d0)
      write(*,*) "delta=",delta

c     boundary condition, Un=1 or uf=1
c     Un=1.d0
c     uf=Un*sqrt(3.d0*Q0)/(2.d0*delta)
      uf=1.d0
      Un=2.d0*uf*delta/sqrt(3.d0*Q0)

c     calculate ln(Q(w)), uxh(w)/Um, uyh(w)/Um arrays
      do p=1,np

      w=(p-1-np/2)*2*wmax/np

      if(p.eq.(np/2+1)) then
        lnQ(p)=log(Q0)
        uxum(p)=0.d0
        uyum(p)=0.d0
        cycle
      end if

      diss=ii*w*b
      w0=1.d0/bk
      kdiss=1.d0/(1.d0-w/w0*ii)
      eia2=exp(-a*ii*w/v/2.d0)
```

```
 eia=eia2**2
 eiai=1.d0/eia
 eia2i=1.d0/eia2
 caw=cos(a*w/v)
 caw2=cos(0.5d0*a*w/v)
 saw=sin(a*w/v)
 saw2=sin(0.5d0*a*w/v)
 kkr=0.75d0*kr+0.25d0*kp
 kkp=0.25d0*kr+0.75d0*kp
 krmkp=kr-1.d0*kp
 aa=16.d0*kp*kr*cos(0.5d0*a*w/v)**2+3.d0*(kr-1.d0*kp)**2
 bb=cos(0.5d0*a*w/v)*(2.d0*(3.d0*kr**2+
$  2.d0*kp*kr+3.d0*kp**2)*cos(a*w/v
$  )+4.d0*(kr+kp)*(m*w**2*kdiss+diss-3.d0*(kr+kp)))
 cc=((-1.d0*kr-1.d0*kp)*(3.d0-1.d0*
$  cos(a*w/v))+m*w**2*kdiss+diss)**2-
$  1.d0*(kp-1.d0*kr)**2*(cos(a*w/v)**2+3.d0*sin(0.5d0*a*w/v)**2)
 z_p=0.5d0*(sqrt(bb**2-4.d0*aa*cc)-1.d0*bb)/aa
 z_m=0.5d0*(-1.d0*sqrt(bb**2-4.d0*aa*cc)-1.d0*bb)/aa
 y_p=sqrt(z_p**2-1.d0)
 y_m=sqrt(z_m**2-1.d0)
 if(real(y_p)*real(z_p).gt.0.d0) then
    y_p=1.d0/(z_p+y_p)
 else
    y_p=1.d0/(z_p-y_p)
 end if
 if(real(y_m)*real(z_m).gt.0.d0) then
    y_m=1.d0/(z_m+y_m)
 else
    y_m=1.d0/(z_m-y_m)
 end if
 if(abs(y_p).gt.1.d0) y_p=1.d0/y_p
 if(abs(y_m).gt.1.d0) y_m=1.d0/y_m


 y_pn=y_p**n
 y_mn=y_m**n
 d_p=caw2*(3.d0*kr+kp)*z_p+
$  m*w**2*kdiss-3.d0*(kr+kp)+2.d0*caw*kp+diss
 d_m=caw2*(3.d0*kr+kp)*z_m+m*w**2*kdiss-
$  3.d0*(kr+kp)+2.d0*caw*kp+diss
 e_p=0.5d0*ii*(kp-1.d0*kr)*saw2*sqrt3*(y_p-1.d0/y_p)
 e_m=0.5d0*ii*(kp-1.d0*kr)*saw2*sqrt3*(y_m-1.d0/y_m)

 v1=d_p**2
 v2=e_m**2
 v3=v1*v2
 v4=-2.d0*d_m*d_p*e_m*e_p
 v5=d_m**2
 v6=e_p**2
 v7=v5*v6
 v8=-v1*v2
 v9=-d_m*d_p*e_m*e_p
 v10=y_m**(n+1)
 v11=-v5*v6
 v12=y_mn**2
 v13=y_pn**2
 v14=1.d0/((6.d0*d_m*d_p*e_m*e_p*y_m*y_mn+
$  2.d0*d_m*d_p*e_m*e_p*v10)*y_p*
$  y_pn+((v11+v9)*y_m*v12+(v9+v8)*v10*
$  y_mn+(v7+v4+v3)*y_m)*y_p+((v
$  7+v9)*y_m*v12+(v9+v3)*v10*y_mn+(v11+v4+v8)*y_m)*y_p*v13)
 v15=-eia2*v1*v2
 v16=d_m*d_p*eia2*e_m*e_p
 v17=y_m**2
 v18=y_m**n
 v19=d_m*d_p*eia2*e_m*e_p*v12
 v20=-eia2*v5*v6
 v21=d_m*d_p*eia2*e_m*e_p*v10*y_mn
 v22=y_p**2
 v23=-d_m*d_p*eia2*e_m*e_p*v10
 v24=-d_m*d_p*eia2*e_m*e_p*y_m*y_mn
 v25=v5*eia2*v6
 v26=v1*eia2*v2
 v27=d_p*eia2*v2*e_p
 v28=-d_m*eia2*e_m*v6
 v29=-d_p*eia2*e_p*v18*v2*y_mn
 v30=-d_p*eia2*e_p*v2
 v31=d_m*eia2*e_m*v6
 v32=d_p*eia2*v2*e_p*v10*y_mn
 v33=d_m*v1*eia2*e_m
 v34=-d_p*eia2*e_p*v5
 v35=v5*d_p*eia2*e_p*v12
 v36=-d_m*eia2*e_m*v1
 v37=v5*d_p*eia2*e_p
 v38=-d_p*eia2*e_p*v12*v5*y_m
 v39=-d_m*d_p*eia2*e_m*e_p
 v40=-d_m*d_p*eia2*e_m*e_p*v18*y_mn
 v41=-d_m*d_p*eia2*e_m*e_p*v12*y_m
 v42=d_m*d_p*eia2*e_m*e_p*v10
 v43=d_m*d_p*eia2*e_m*e_p*y_m*y_mn

 f=-v14*(((-2.d0*d_m*d_p*eia2*e_m*e_p*v17-
$  2.d0*d_m*d_p*eia2*e_m
$  *e_p)*y_mn*y_p+v24+v23+(v24+v23)*v22)*
$  y_pn+v13*((-eia2*v1*v
$  18*v2*y_mn+v19+(v16+v26)*v17)*y_p-eia2*v12*v5*v6*y_m+(v25+v
$  16)*y_m+v21)+(v19+v1*eia2*v2*v18*y_mn+(v16+v15)*v17
$  )*y_p+(v5*ei
$  a2*v6*y_m*v12+v21+(v20+v16)*y_m)*v22)
 g=v14*(((2.d0*d_p*eia2*e_p*v2-
$  2.d0*d_p*eia2*e_p*v17*v2)*y_mn*y_p+v
$  22*(-d_m*eia2*e_m*v6*y_m*y_mn-d_m*eia2*e_m*v10*v6)+d_m*
$  eia2*e_m*v6*y_m*y_mn+d_m*eia2*e_m*v6*v10)*
$  y_pn+v13*((d_m*eia2*e
$  _m*v6*v12+v29+(v31+v27)*v17)*y_p-d_m*eia2*e_m*v12*v6*y_m+(v
$  28+v30)*y_m+v32)+(-d_m*eia2*e_m*v12*v6+v29+(v28+v27)*v17)*y
$  _p+(d_m*eia2*e_m*v6*y_m*v12+v32+(v31+v30)*y_m)*v22)
 hh=-v14*(((2.d0*d_p*eia2*e_p*v17*v5-
$  2.d0*d_p*eia2*e_p*v5)*y_mn
$  *y_p-d_m*eia2*e_m*v1*y_m*y_mn+(d_m*v1*eia2*e_m*y_m*y_mn+d_m
     $  *v1*eia2*e_m*v10)*v22-d_m*eia2*e_m*v1*v10)*y_pn+v13*((-
     $  d_m*eia2*e_m*v1*v18*y_mn+v35+(v34+v36)*v17)*y_p+d_m*
     $  v1*eia2*e_m
     $  *v10*y_mn+(v37+v33)*y_m+v38)+
     $  (v35+d_m*v1*eia2*e_m*v18*y_mn+(v34
     $  +v33)*v17)*y_p+v22*(-d_m*eia2*e_m*v1*v10*y_mn+(v37+v36)*y_m
     $  +v38))
      i=v14*(((2.d0*d_m*d_p*eia2*e_m*e_p*v17+
     $  2.d0*d_m*d_p*eia2*e_m*e_p)*
     $  y_mn*y_p+v43+v42+(v43+v42)*v22)*y_pn+(-eia2*v12*v5*v6+v40+(
     $  v25+v39)*v17)*y_p+v22*(-eia2*v1*v10*v2*y_mn+(v39+v26)*y_m+v
     $  41)+((v5*eia2*v6*v12+v40+(v20+v39)*v17)*
     $  y_p+v41+v1*eia2*v2*v10*
     $  y_mn+(v39+v15)*y_m)*v13)

      j=0.25d0*(4.d0*m*w**2*kdiss+
     $  (1.d0-eiai)*hh*krmkp*sqrt3+(8.d0*caw-8.d0)*k
     $  p+((4.d0*eiai+4.d0)*f-8.d0)*kkr+4.d0*diss)
      k=-0.25d0*((eiai-1.0)*i*krmkp*sqrt3+(-4.d0*eiai-4.d0)*g*kkr)
      l=-0.25d0*((eiai-1.d0)*f*krmkp*sqrt3+(-4.d0*eiai-4.d0)*hh*kkp)
      mm=0.25d0*(4.0*m*w**2*kdiss+
     $  (1.d0-1.d0*eiai)*g*krmkp*sqrt3+(8.d0*caw-8.d0)*k
     $  r+((4.d0*eiai+4.d0)*i-8.d0)*kkp+4.d0*diss)
      q=-(2.d0*j*mm-2.d0*k*l)/((ii*kpi*l-ii*k*kpi)
     $  *sqrt3*saw2+(j*kpi*sqrt3**2-kpi*mm)*caw2
     $  +j*kpi*sqrt3**2+(kpi-2.d0*j)*mm+2.d0*k*l)

      uxumw=(3.d0*(1.d0+eia2)*k*kpi+
     $  sqrt3*kpi*(mm-eia2*mm))/(2.d0*eia2*(k*l-j*mm))
      uyumw=-(3.d0*(1.d0+eia2)*j*kpi+
     $  sqrt3*kpi*(l-eia2*l))/(2.d0*eia2*(k*l-j*mm))

c     system for components above crack line
      if(h.gt.1.d0) then
        w1=y_m**n
        w2=y_p**n
        w3=w1**2
```

```
   w4=w2**2
   w5=y_m**(h-0.5d0)
   w6=y_p**(h-0.5d0)
   w7=y_m**(n-h+0.5d0)
   w8=y_p**(n-h+0.5d0)
   w9=w1*w7
   w10=w2*w8
   w11=y_p**(n-h-0.5d0)
   w12=y_m**(n+h-0.5d0)
   w13=(-(d_m*e_p*(1.d0+w1)*(-1.d0+w2))+
$    d_p*e_m*(-1.d0+w1)*(1.d0+w2))*
$    (d_p*e_m*(1.d0+w1)*(-1.d0+w2)-d_m*e_p*(-1.d0+w1)*(1.d0+w2))

   uxumw=-((uxumw*v5*v6*(-1.d0+w3)*(-w10+w6)+
$    d_p*v2*(d_p*uxumw*(-1.d0+w4)*(w5-w9)+
$    e_p*uyumw*(w10+2.d0*w12*w2-(1.d0+w4)*w5+w6-w3*(w10+w6)-
$    2.d0*w2*w7+(1.d0+w4)*w9))+
$    d_m*e_m*e_p*(-(e_p*uyumw*
$    (-w10+(-1.d0+w4)*w5+w6+(-1.d0+w4)*w9+
$    (-w10+w6)*y_m**(2.d0*n)-2.d0*w1*w11*(-y_p+y_p**(2.d0*h))))+
$    d_p*uxumw*(w10-2.d0*w12*w2+(1.d0+w4)*w5+w6+w3*(w10+w6)-
$    2.d0*w2*w7+(1.d0+w4)*w9-2.d0*w1*w11*(y_p+y_p**(2.d0*h)))))/
$    (exp((ii*gn*w)/(2.d0*v))*w13))

   uyumw=-((uyumw*v1*v2*(-1.d0+w3)*(-w10+w6)+
$    e_p*v5*(e_p*uyumw*(-1.d0+w4)*(w5-w9)+
$    d_p*uxumw*(w10+2.d0*w12*w2-(1.d0+w4)*w5+w6-w3*(w10+w6)-
$    2.d0*w2*w7+(1.d0+w4)*w9))+
$    d_m*d_p*e_m*(-(d_p*uxumw*
$    (-w10+(-1.d0+w4)*w5+w6+w3*(-w10+w6)+(-1.d0+w4)*w9-
$    2.d0*w1*w11*(-y_p+y_p**(2.d0*h))))-
$    e_p*uyumw*(-w10+2.d0*w12*w2-(1.d0+w4)*w5-w6-w3*(w10+w6)+
$    2.d0*w2*w7-(1.d0+w4)*w9+2.d0*w1*w11*(y_p+y_p**(2.d0*h)))))/
$    (exp((ii*gn*w)/(2.d0*v))*w13))
 end if

 lnQ(p)=log(Q)
```

```
      uxum(p)=uxumw
      uyum(p)=uyumw

      end do

c     inverse FT of ln(Q(w))
      do p=1,np
        lnQ(p)=lnQ(p)*(-1)**(p-1)
      end do

      call dfftw_plan_dft_1d_(planLNQT,np,lnQ,lnQt,
     &  FFTW_FORWARD,FFTW_ESTIMATE)
      call dfftw_execute_(planLNQT)

      do p=1,np
        lnQt(p)=lnQt(p)*dw/(2.d0*pi)*(-1)**(p-1)
      end do

c     Qm(w) calculation
      do p=(np/2+2),np
        lnQt(p)=0.d0
      end do

      do p=1,np
        lnQt(p)=lnQt(p)*(-1)**(p-1)
      end do

      call dfftw_plan_dft_1d_(planQM,np,lnQt,Qm,
     &  FFTW_BACKWARD,FFTW_ESTIMATE)
      call dfftw_execute_(planQM)

      do p=1,np
        Qm(p)=exp(Qm(p)*dt*(-1)**(p-1))
      end do

c     W(-i/bk) for Kelvin dissipation
      pole=bk*Un*sqrt(Q0)/delta
```

```
c     Um(w) calculation
      do p=1,np
        if(p.eq.(np/2+1)) then
          Um(p)=0.d0
          cycle
        end if
        w=-wmax+(p-1)*dw
        Um(p)=(-pole-(ii*Un*Q0*Qm(p))/(Qm(np/2+1)*w))/(1.d0-ii*bk*w)
      end do

c     ux(w), uy(w) arrays
      do p=1,np
        ux(p)=Um(p)*uxum(p)
        uy(p)=Um(p)*uyum(p)
      end do

c     coefficient of 1/(iw) behavior of uy(w) at w=0
      uyw0=(Q0-Un)*(1.d0-(h-0.5d0)/n)

c     subtraction of of 1/(iw) behavior
      do p=1,np
        if(p.eq.(np/2+1)) cycle
        w=-wmax+(p-1)*dw
        uy(p)=uy(p)-uyw0/(ii*w*(1.d0+ii*w))
      end do

c     correction at w=0 using quadratic approximation
      ux(np/2+1)=(-ux(np/2-1)+3.d0*ux(np/2)+ux(np/2+2))/3.d0
      uy(np/2+1)=(-uy(np/2-1)+3.d0*uy(np/2)+uy(np/2+2))/3.d0

c     inverse FT of ux(w), uy(w)
      do p=1,np
        ux(p)=ux(p)*(-1)**(p-1)
        uy(p)=uy(p)*(-1)**(p-1)
      end do
      call dfftw_plan_dft_1d_(planUXT,np,ux,uxt,
     &  FFTW_FORWARD,FFTW_ESTIMATE)
      call dfftw_execute_(planUXT)

      call dfftw_plan_dft_1d_(planUYT,np,uy,uyt,
     &  FFTW_FORWARD,FFTW_ESTIMATE)
      call dfftw_execute_(planUYT)

      do p=1,np
        uxt(p)=uxt(p)*dw/(2.d0*pi)*(-1)**(p-1)
        uyt(p)=uyt(p)*dw/(2.d0*pi)*(-1)**(p-1)
      end do

c     add back analytical subtraction
      do p=1,(np/2+1)
        uyt(p)=uyt(p)+Un+uyw0*(1.d0-exp(-tmax+(p-1)*dt))
      end do

      do p=(np/2+2),np
        uyt(p)=uyt(p)+Un
      end do

c     check that U(0)=uf=1
c     t0=a/(2.d0*v)
c     n0=floor(t0/dt)+(np/2+1)
c     ux0=uxt(n0)+(uxt(n0+1)-uxt(n0))*(t0/dt-floor(t0/dt))
c     uy0=uyt(n0)+(uyt(n0+1)-uyt(n0))*(t0/dt-floor(t0/dt))
c     u0=-(ux0-uxt(np/2+1))/(2.d0*sqrt3)+(uy0-uyt(np/2+1))/2.d0

c     compare delta estimate with direct integration
c     dtest=Un*sqrt(3.d0*Q0)/(2.d0*realpart(u0))
c     write(*,*) "dtest=",dtest
c     write(*,*) "%diff=",100*(delta-dtest)/delta

c     print t,ux,uy to a text file
      open(unit=10,file="t.txt",action="write")
      open(unit=20,file="ux.txt",action="write")
      open(unit=30,file="uy.txt",action="write")

      do p=1,np,np/nplot
        write(10,*) -tmax+(p-1)*dt
        write(20,*) realpart(uxt(p))
        write(30,*) realpart(uyt(p))
      end do
c     boundary condition Un
      write(30,*) Un

      close(10)
      close(20)
      close(30)

      call dfftw_destroy_plan(planLNQT)
      call dfftw_destroy_plan(planQM)
      call dfftw_destroy_plan(planUXT)
      call dfftw_destroy_plan(planUYT)

      end


c     qcomp from kelvin_integrate
      real*8 function qcomp(s)
      implicit complex*16 (a-h,o-z)
      common/a/n,m,a,b,bk,kp,kr,kpi,v

      real*8 n,m,a,b,bk,kp,kr,kpi,v
      real*8 s,w,kkr,kkp,krmkp,dw_ds,wfac
      complex*16 ii,i,j,k,l,mm,kdiss,logq,ln_q_diff

      pi=3.1415926535897931
      ii=cmplx(0.d0,1.d0)
      sqrt3=sqrt(3.d0)

      if(s.eq.0) s=1.d-05
```

```
 wfac=10.d0
 w=wfac*s/dsqrt(1.d0-s*s)
 dw_ds=wfac/dsqrt(1.d0-s*s)/(1.d0-s*s)

 diss=ii*w*b
 w0=1.d0/bk
 kdiss=1.d0/(1.d0-w/w0*ii)
 eia2=exp(-a*ii*w/v/2.d0)
 eia=eia2**2
 eiai=1.d0/eia
 eia2i=1.d0/eia2
 caw=cos(a*w/v)
 caw2=cos(0.5d0*a*w/v)
 saw=sin(a*w/v)
 saw2=sin(0.5d0*a*w/v)
 kkr=0.75d0*kr+0.25d0*kp
 kkp=0.25d0*kr+0.75d0*kp
 krmkp=kr-1.d0*kp
 aa=16.d0*kp*kr*cos(0.5d0*a*w/v)**2+3.d0*(kr-1.d0*kp)**2
 bb=cos(0.5d0*a*w/v)*(2.d0*(3.d0*kr**2+
$  2.d0*kp*kr+3.d0*kp**2)*cos(a*w/v
$  )+4.d0*(kr+kp)*(m*w**2*kdiss+diss-3.d0*(kr+kp)))
 cc=((-1.d0*kr-1.d0*kp)*(3.d0-1.d0*
$  cos(a*w/v))+m*w**2*kdiss+diss)**2-
$  1.d0*(kp-1.d0*kr)**2*(cos(a*w/v)**2+3.d0*sin(0.5d0*a*w/v)**2)
 z_p=0.5d0*(sqrt(bb**2-4.d0*aa*cc)-1.d0*bb)/aa
 z_m=0.5d0*(-1.d0*sqrt(bb**2-4.d0*aa*cc)-1.d0*bb)/aa
 y_p=sqrt(z_p**2-1.d0)
 y_m=sqrt(z_m**2-1.d0)
 if(real(y_p)*real(z_p).gt.0.d0) then
    y_p=1.d0/(z_p+y_p)
 else
    y_p=1.d0/(z_p-y_p)
 end if
 if(real(y_m)*real(z_m).gt.0.d0) then
    y_m=1.d0/(z_m+y_m)
 else
```

```
         y_m=1.d0/(z_m-y_m)
      end if
      if(abs(y_p).gt.1.d0) y_p=1.d0/y_p
      if(abs(y_m).gt.1.d0) y_m=1.d0/y_m


      y_pn=y_p**n
      y_mn=y_m**n
      d_p=caw2*(3.d0*kr+kp)*z_p+
     $  m*w**2*kdiss-3.d0*(kr+kp)+2.d0*caw*kp+diss
      d_m=caw2*(3.d0*kr+kp)*z_m+m*w**2*kdiss-
     $  3.d0*(kr+kp)+2.d0*caw*kp+diss
      e_p=0.5d0*ii*(kp-1.d0*kr)*saw2*sqrt3*(y_p-1.d0/y_p)
      e_m=0.5d0*ii*(kp-1.d0*kr)*saw2*sqrt3*(y_m-1.d0/y_m)

      v1=d_p**2
      v2=e_m**2
      v3=v1*v2
      v4=-2.d0*d_m*d_p*e_m*e_p
      v5=d_m**2
      v6=e_p**2
      v7=v5*v6
      v8=-v1*v2
      v9=-d_m*d_p*e_m*e_p
      v10=y_m**(n+1)
      v11=-v5*v6
      v12=y_mn**2
      v13=y_pn**2
      v14=1.d0/((6.d0*d_m*d_p*e_m*e_p*y_m*y_mn+
     $  2.d0*d_m*d_p*e_m*e_p*v10)*y_p*
     $  y_pn+((v11+v9)*y_m*v12+(v9+v8)*v10*
     $  y_mn+(v7+v4+v3)*y_m)*y_p+((v
     $  7+v9)*y_m*v12+(v9+v3)*v10*y_mn+(v11+v4+v8)*y_m)*y_p*v13)
      v15=-eia2*v1*v2
      v16=d_m*d_p*eia2*e_m*e_p
      v17=y_m**2
      v18=y_m**n
```

```
 v19=d_m*d_p*eia2*e_m*e_p*v12
 v20=-eia2*v5*v6
 v21=d_m*d_p*eia2*e_m*e_p*v10*y_mn
 v22=y_p**2
 v23=-d_m*d_p*eia2*e_m*e_p*v10
 v24=-d_m*d_p*eia2*e_m*e_p*y_m*y_mn
 v25=v5*eia2*v6
 v26=v1*eia2*v2
 v27=d_p*eia2*v2*e_p
 v28=-d_m*eia2*e_m*v6
 v29=-d_p*eia2*e_p*v18*v2*y_mn
 v30=-d_p*eia2*e_p*v2
 v31=d_m*eia2*e_m*v6
 v32=d_p*eia2*v2*e_p*v10*y_mn
 v33=d_m*v1*eia2*e_m
 v34=-d_p*eia2*e_p*v5
 v35=v5*d_p*eia2*e_p*v12
 v36=-d_m*eia2*e_m*v1
 v37=v5*d_p*eia2*e_p
 v38=-d_p*eia2*e_p*v12*v5*y_m
 v39=-d_m*d_p*eia2*e_m*e_p
 v40=-d_m*d_p*eia2*e_m*e_p*v18*y_mn
 v41=-d_m*d_p*eia2*e_m*e_p*v12*y_m
 v42=d_m*d_p*eia2*e_m*e_p*v10
 v43=d_m*d_p*eia2*e_m*e_p*y_m*y_mn

 f=-v14*(((-2.d0*d_m*d_p*eia2*e_m*e_p*v17-
$  2.d0*d_m*d_p*eia2*e_m
$  *e_p)*y_mn*y_p+v24+v23+(v24+v23)*v22)*
$  y_pn+v13*((-eia2*v1*v
$  18*v2*y_mn+v19+(v16+v26)*v17)*y_p-eia2*v12*v5*v6*y_m+(v25+v
$  16)*y_m+v21)+(v19+v1*eia2*v2*v18*y_mn+(v16+v15)*v17
$  )*y_p+(v5*ei
$  a2*v6*y_m*v12+v21+(v20+v16)*y_m)*v22)
 g=v14*(((2.d0*d_p*eia2*e_p*v2-
$  2.d0*d_p*eia2*e_p*v17*v2)*y_mn*y_p+v
$  22*(-d_m*eia2*e_m*v6*y_m*y_mn-d_m*eia2*e_m*v10*v6)+d_m*
$  eia2*e_m*v6*y_m*y_mn+d_m*eia2*e_m*v6*v10)*
$  y_pn+v13*((d_m*eia2*e
$  _m*v6*v12+v29+(v31+v27)*v17)*y_p-d_m*eia2*e_m*v12*v6*y_m+(v
$  28+v30)*y_m+v32)+(-d_m*eia2*e_m*v12*v6+v29+(v28+v27)*v17)*y
$  _p+(d_m*eia2*e_m*v6*y_m*v12+v32+(v31+v30)*y_m)*v22)
 hh=-v14*(((2.d0*d_p*eia2*e_p*v17*v5-
$  2.d0*d_p*eia2*e_p*v5)*y_mn
$  *y_p-d_m*eia2*e_m*v1*y_m*y_mn+(d_m*v1*eia2*e_m*y_m*y_mn+d_m
$  *v1*eia2*e_m*v10)*v22-d_m*eia2*e_m*v1*v10)*y_pn+v13*((-
$  d_m*eia2*e_m*v1*v18*y_mn+v35+(v34+v36)*v17)*y_p+d_m*
$  v1*eia2*e_m
$  *v10*y_mn+(v37+v33)*y_m+v38)+
$  (v35+d_m*v1*eia2*e_m*v18*y_mn+(v34
$  +v33)*v17)*y_p+v22*(-d_m*eia2*e_m*v1*v10*y_mn+(v37+v36)*y_m
$  +v38))
 i=v14*(((2.d0*d_m*d_p*eia2*e_m*e_p*v17+
$  2.d0*d_m*d_p*eia2*e_m*e_p)*
$  y_mn*y_p+v43+v42+(v43+v42)*v22)*y_pn+(-eia2*v12*v5*v6+v40+(
$  v25+v39)*v17)*y_p+v22*(-eia2*v1*v10*v2*y_mn+(v39+v26)*y_m+v
$  41)+((v5*eia2*v6*v12+v40+(v20+v39)*v17)*
$  y_p+v41+v1*eia2*v2*v10*
$  y_mn+(v39+v15)*y_m)*v13)

 j=0.25d0*(4.d0*m*w**2*kdiss+
$  (1.d0-eiai)*hh*krmkp*sqrt3+(8.d0*caw-8.d0)*k
$  p+((4.d0*eiai+4.d0)*f-8.d0)*kkr+4.d0*diss)
 k=-0.25d0*((eiai-1.0)*i*krmkp*sqrt3+(-4.d0*eiai-4.d0)*g*kkr)
 l=-0.25d0*((eiai-1.d0)*f*krmkp*sqrt3+(-4.d0*eiai-4.d0)*hh*kkp)
 mm=0.25d0*(4.0*m*w**2*kdiss+
$  (1.d0-1.d0*eiai)*g*krmkp*sqrt3+(8.d0*caw-8.d0)*k
$  r+((4.d0*eiai+4.d0)*i-8.d0)*kkp+4.d0*diss)
 q=-(2.d0*j*mm-2.d0*k*l)/((ii*kpi*l-ii*k*kpi)
$  *sqrt3*saw2+(j*kpi*sqrt3**2-kpi*mm)*caw2
$  +j*kpi*sqrt3**2+(kpi-2.d0*j)*mm+2.d0*k*l)

 logq=zlog(q)
 ln_q_diff=logq-conjg(logq)

 ctmp=-dw_ds/(2.0*pi)*
$  (
$    ln_q_diff*
$      (
$        1.d0/(2.d0*ii*w)+ii*w/(2.d0*(w0**2+w**2))
$      )
$    +w0*zlog(q*conjg(q))/(2.d0*(w0**2+w**2))
$  )
 qcomp=realpart(ctmp)
 return
 end
```

Appendix B

Energy flux script

This is the Fortran script we wrote to compute the lattice energy flux vector from Chapter 5. It takes the components at each atom in a region of the lattice and returns the energy flux vectors averaged over a period. The input parameters are the number of rows and columns in the region, and the number of time points to be averaged. Another version of the component script from Appendix A is used to find the components at each atom in the region.

```
      program energy

      implicit real*8 (a-z)
      integer i,j,r,c,t,nrow,ncol,nt,col,natom,atom,nl
      parameter(nrow=20,ncol=100,nt=20,col=2*ncol+1,
     &  natom=(nrow+1)*col*nt,atom=2*nrow*(col-2)*nt,
     &  nl=2*nrow*(col-2))

      real*8 above(natom,4),below(natom,4)
      real*8 atoms(atom,4),latt(nl,4),olat(nl,4)
      real*8 len(nl),loglen(nl)

c     read coordinates, velocities
      open(unit=10,file="above.txt",action="read")
      open(unit=20,file="below.txt",action="read")

      do i=1,natom
        read(10,*) above(i,1),above(i,2),above(i,3),above(i,4)
```

```
          read(20,*) below(i,1),below(i,2),below(i,3),below(i,4)
        end do

        close(10)
        close(20)

c       lattice constants
        sqrt3=sqrt(3.d0)
        a=1.d0
        a2=0.5d0*a
        ah=0.5d0*sqrt3*a
        kp=1.d0

c       atom array index
        j=1

c       six direction unit vectors (parallel only)
        e1x=-0.5d0
        e1y=sqrt3/2.d0
        e2x=0.5d0
        e2y=sqrt3/2.d0
        e3x=-0.5d0
        e3y=-sqrt3/2.d0
        e4x=0.5d0
        e4y=-sqrt3/2.d0
        e5x=-1.d0
        e5y=0.d0
        e6x=1.d0
        e6y=0.d0

c       loop over crack line n=1/2
        do c=nt+1,2*ncol*nt
c         displacements along all six directions
          d1x=above(c+col*nt,1)-above(c,1)
          d1y=above(c+col*nt,2)-above(c,2)
          d2x=above(c+(col+1)*nt,1)-above(c,1)
          d2y=above(c+(col+1)*nt,2)-above(c,2)
          d3x=below(c,1)-above(c,1)
          d3y=below(c,2)-above(c,2)
          d4x=below(c+1*nt,1)-above(c,1)
          d4y=below(c+1*nt,2)-above(c,2)
          d5x=above(c-1*nt,1)-above(c,1)
          d5y=above(c-1*nt,2)-above(c,2)
          d6x=above(c+1*nt,1)-above(c,1)
          d6y=above(c+1*nt,2)-above(c,2)
  c       set broken bond displacments to 0
          if(c.le.ncol*nt) then
            d3x=0.d0
            d3y=0.d0
            d4x=0.d0
            d4y=0.d0
          end if
  c       displacement dot products dj.ej
  c       force along each direction
          d1e1=d1x*e1x+d1y*e1y
          f1x=kp*e1x*d1e1
          f1y=kp*e1y*d1e1
          d2e2=d2x*e2x+d2y*e2y
          f2x=kp*e2x*d2e2
          f2y=kp*e2y*d2e2
          d3e3=d3x*e3x+d3y*e3y
          f3x=kp*e3x*d3e3
          f3y=kp*e3y*d3e3
          d4e4=d4x*e4x+d4y*e4y
          f4x=kp*e4x*d4e4
          f4y=kp*e4y*d4e4
          d5e5=d5x*e5x+d5y*e5y
          f5x=kp*e5x*d5e5
          f5y=kp*e5y*d5e5
          d6e6=d6x*e6x+d6y*e6y
          f6x=kp*e6x*d6e6
          f6y=kp*e6y*d6e6
  c       velocity components
          vx=above(c,3)
          vy=above(c,4)
  c       dot products fj.v
  c       energy flux components in each direction
          f1v=f1x*vx+f1y*vy
          p1x=e1x*f1v
          p1y=e1y*f1v
          f2v=f2x*vx+f2y*vy
          p2x=e2x*f2v
          p2y=e2y*f2v
          f3v=f3x*vx+f3y*vy
          p3x=e3x*f3v
          p3y=e3y*f3v
          f4v=f4x*vx+f4y*vy
          p4x=e4x*f4v
          p4y=e4y*f4v
          f5v=f5x*vx+f5y*vy
          p5x=e5x*f5v
          p5y=e5y*f5v
          f6v=f6x*vx+f6y*vy
          p6x=e6x*f6v
          p6y=e6y*f6v
  c       total energy flux
          px=p1x+p2x+p3x+p4x+p5x+p6x
          py=p1y+p2y+p3y+p4y+p5y+p6y
  c       coordinates of current atom
          x=(ceiling(1.d0*c/nt)-ncol-1)*a+above(c,1)
          y=above(c,2)
  c       fill atoms array with coordinates, flux vector components
          atoms(j,1)=x
          atoms(j,2)=y
          atoms(j,3)=px
          atoms(j,4)=py
          j=j+1
        end do

  c     loop over crack line n=-1/2
        do c=nt+1,2*ncol*nt
```

```
c       displacements along all six directions
        d1x=above(c-1*nt,1)-below(c,1)
        d1y=above(c-1*nt,2)-below(c,2)
        d2x=above(c,1)-below(c,1)
        d2y=above(c,2)-below(c,2)
        d3x=below(c+(col-1)*nt,1)-below(c,1)
        d3y=below(c+(col-1)*nt,2)-below(c,2)
        d4x=below(c+col*nt,1)-below(c,1)
        d4y=below(c+col*nt,2)-below(c,2)
        d5x=below(c-1*nt,1)-below(c,1)
        d5y=below(c-1*nt,2)-below(c,2)
        d6x=below(c+1*nt,1)-below(c,1)
        d6y=below(c+1*nt,2)-below(c,2)
c       set broken bond displacments to 0
        if(c.le.ncol*nt) then
          d1x=0.d0
          d1y=0.d0
          d2x=0.d0
          d2y=0.d0
        end if
c       displacement dot products dj.ej
c       force along each direction
        d1e1=d1x*e1x+d1y*e1y
        f1x=kp*e1x*d1e1
        f1y=kp*e1y*d1e1
        d2e2=d2x*e2x+d2y*e2y
        f2x=kp*e2x*d2e2
        f2y=kp*e2y*d2e2
        d3e3=d3x*e3x+d3y*e3y
        f3x=kp*e3x*d3e3
        f3y=kp*e3y*d3e3
        d4e4=d4x*e4x+d4y*e4y
        f4x=kp*e4x*d4e4
        f4y=kp*e4y*d4e4
        d5e5=d5x*e5x+d5y*e5y
        f5x=kp*e5x*d5e5
        f5y=kp*e5y*d5e5
        d6e6=d6x*e6x+d6y*e6y
        f6x=kp*e6x*d6e6
        f6y=kp*e6y*d6e6
c       velocity components
        vx=below(c,3)
        vy=below(c,4)
c       dot products fj.v
c       energy flux components in each direction
        f1v=f1x*vx+f1y*vy
        p1x=e1x*f1v
        p1y=e1y*f1v
        f2v=f2x*vx+f2y*vy
        p2x=e2x*f2v
        p2y=e2y*f2v
        f3v=f3x*vx+f3y*vy
        p3x=e3x*f3v
        p3y=e3y*f3v
        f4v=f4x*vx+f4y*vy
        p4x=e4x*f4v
        p4y=e4y*f4v
        f5v=f5x*vx+f5y*vy
        p5x=e5x*f5v
        p5y=e5y*f5v
        f6v=f6x*vx+f6y*vy
        p6x=e6x*f6v
        p6y=e6y*f6v
c       total energy flux
        px=p1x+p2x+p3x+p4x+p5x+p6x
        py=p1y+p2y+p3y+p4y+p5y+p6y
c       coordinates of current atom
        x=(ceiling(1.d0*c/nt)-ncol-1)*a-a2+below(c,1)
        y=-ah+below(c,2)
c       fill atoms array with coordinates, flux vector components
        atoms(j,1)=x
        atoms(j,2)=y
        atoms(j,3)=px
        atoms(j,4)=py
        j=j+1
      end do

c     loops over atom rows above crack line
      do r=1,nrow-1
      h=0.5d0+r
      gn=mod(int(h-0.5d0),2)
c     loop over columns
      do c=(1+r*col)*nt+1,(2*ncol+r*col)*nt
c       displacements along all six directions
        d1x=above(c+(col-gn)*nt,1)-above(c,1)
        d1y=above(c+(col-gn)*nt,2)-above(c,2)
        d2x=above(c+(col-gn+1)*nt,1)-above(c,1)
        d2y=above(c+(col-gn+1)*nt,2)-above(c,2)
        d3x=above(c-(col+gn)*nt,1)-above(c,1)
        d3y=above(c-(col+gn)*nt,2)-above(c,2)
        d4x=above(c-(col+gn-1)*nt,1)-above(c,1)
        d4y=above(c-(col+gn-1)*nt,2)-above(c,2)
        d5x=above(c-1*nt,1)-above(c,1)
        d5y=above(c-1*nt,2)-above(c,2)
        d6x=above(c+1*nt,1)-above(c,1)
        d6y=above(c+1*nt,2)-above(c,2)
c       displacement dot products dj.ej
c       force along each direction
        d1e1=d1x*e1x+d1y*e1y
        f1x=kp*e1x*d1e1
        f1y=kp*e1y*d1e1
        d2e2=d2x*e2x+d2y*e2y
        f2x=kp*e2x*d2e2
        f2y=kp*e2y*d2e2
        d3e3=d3x*e3x+d3y*e3y
        f3x=kp*e3x*d3e3
        f3y=kp*e3y*d3e3
        d4e4=d4x*e4x+d4y*e4y
        f4x=kp*e4x*d4e4
        f4y=kp*e4y*d4e4
        d5e5=d5x*e5x+d5y*e5y
```

```
         f5x=kp*e5x*d5e5
         f5y=kp*e5y*d5e5
         d6e6=d6x*e6x+d6y*e6y
         f6x=kp*e6x*d6e6
         f6y=kp*e6y*d6e6
c        velocity components
         vx=above(c,3)
         vy=above(c,4)
c        dot products fj.v
c        energy flux components in each direction
         f1v=f1x*vx+f1y*vy
         p1x=e1x*f1v
         p1y=e1y*f1v
         f2v=f2x*vx+f2y*vy
         p2x=e2x*f2v
         p2y=e2y*f2v
         f3v=f3x*vx+f3y*vy
         p3x=e3x*f3v
         p3y=e3y*f3v
         f4v=f4x*vx+f4y*vy
         p4x=e4x*f4v
         p4y=e4y*f4v
         f5v=f5x*vx+f5y*vy
         p5x=e5x*f5v
         p5y=e5y*f5v
         f6v=f6x*vx+f6y*vy
         p6x=e6x*f6v
         p6y=e6y*f6v
c        total energy flux
         px=p1x+p2x+p3x+p4x+p5x+p6x
         py=p1y+p2y+p3y+p4y+p5y+p6y
c        coordinates of current atom
         x=(ceiling(1.d0*c/nt)-ncol-1-r*col)*a-gn*a2+above(c,1)
         y=r*ah+above(c,2)
c        fill atoms array with coordinates, flux vector components
         atoms(j,1)=x
         atoms(j,2)=y
```

```
        atoms(j,3)=px
        atoms(j,4)=py
        j=j+1
      end do
      end do

c     loops over atom rows below crack line
      do r=1,nrow-1
      h=0.5d0+r
      gnn=mod(int(h+0.5d0),2)
c     loop over columns
      do c=(1+r*col)*nt+1,(2*ncol+r*col)*nt
c       displacements along all six directions
        d1x=below(c-(col+gnn)*nt,1)-below(c,1)
        d1y=below(c-(col+gnn)*nt,2)-below(c,2)
        d2x=below(c-(col+gnn-1)*nt,1)-below(c,1)
        d2y=below(c-(col+gnn-1)*nt,2)-below(c,2)
        d3x=below(c+(col-gnn)*nt,1)-below(c,1)
        d3y=below(c+(col-gnn)*nt,2)-below(c,2)
        d4x=below(c+(col-gnn+1)*nt,1)-below(c,1)
        d4y=below(c+(col-gnn+1)*nt,2)-below(c,2)
        d5x=below(c-1*nt,1)-below(c,1)
        d5y=below(c-1*nt,2)-below(c,2)
        d6x=below(c+1*nt,1)-below(c,1)
        d6y=below(c+1*nt,2)-below(c,2)
c       displacement dot products dj.ej
c       force along each direction
        d1e1=d1x*e1x+d1y*e1y
        f1x=kp*e1x*d1e1
        f1y=kp*e1y*d1e1
        d2e2=d2x*e2x+d2y*e2y
        f2x=kp*e2x*d2e2
        f2y=kp*e2y*d2e2
        d3e3=d3x*e3x+d3y*e3y
        f3x=kp*e3x*d3e3
        f3y=kp*e3y*d3e3
        d4e4=d4x*e4x+d4y*e4y
        f4x=kp*e4x*d4e4
        f4y=kp*e4y*d4e4
        d5e5=d5x*e5x+d5y*e5y
        f5x=kp*e5x*d5e5
        f5y=kp*e5y*d5e5
        d6e6=d6x*e6x+d6y*e6y
        f6x=kp*e6x*d6e6
        f6y=kp*e6y*d6e6
c       velocity components
        vx=below(c,3)
        vy=below(c,4)
c       dot products fj.v
c       energy flux components in each direction
        f1v=f1x*vx+f1y*vy
        p1x=e1x*f1v
        p1y=e1y*f1v
        f2v=f2x*vx+f2y*vy
        p2x=e2x*f2v
        p2y=e2y*f2v
        f3v=f3x*vx+f3y*vy
        p3x=e3x*f3v
        p3y=e3y*f3v
        f4v=f4x*vx+f4y*vy
        p4x=e4x*f4v
        p4y=e4y*f4v
        f5v=f5x*vx+f5y*vy
        p5x=e5x*f5v
        p5y=e5y*f5v
        f6v=f6x*vx+f6y*vy
        p6x=e6x*f6v
        p6y=e6y*f6v
c       total energy flux
        px=p1x+p2x+p3x+p4x+p5x+p6x
        py=p1y+p2y+p3y+p4y+p5y+p6y
c       coordinates of current atom
        x=(ceiling(1.d0*c/nt)-ncol-1-r*col)*a-gnn*a2+below(c,1)
        y=-(r+1)*ah+below(c,2)
```

```
c       fill atoms array with coordinates, flux vector components
        atoms(j,1)=x
        atoms(j,2)=y
        atoms(j,3)=px
        atoms(j,4)=py
        j=j+1
      end do
      end do

c     initialize latt, olat arrays
      do j=1,nl
        latt(j,1)=0.d0
        latt(j,2)=0.d0
        latt(j,3)=0.d0
        latt(j,4)=0.d0
        olat(j,1)=0.d0
        olat(j,2)=0.d0
        olat(j,3)=0.d0
        olat(j,4)=0.d0
      end do

c     fill latt array with time averages
      do j=1,nl
        do t=1,nt
          latt(j,1)=latt(j,1)+atoms(j*nt-(nt-t),1)/nt
          latt(j,2)=latt(j,2)+atoms(j*nt-(nt-t),2)/nt
          latt(j,3)=latt(j,3)+atoms(j*nt-(nt-t),3)/nt
          latt(j,4)=latt(j,4)+atoms(j*nt-(nt-t),4)/nt
        end do
      end do

c     fill len array with flux vector magnitudes
      do j=1,nl
        len(j)=sqrt(latt(j,3)**2+latt(j,4)**2)
      end do

c     fill loglen array
```

```
      do j=1,nl
        loglen(j)=log10(len(j))
      end do

c     minimum, maximum log magnitudes
      minlen=minval(loglen)
      maxlen=maxval(loglen)

c     scale log magnitudes between [0,1]
      do j=1,nl
        loglen(j)=(loglen(j)-minlen)/(maxlen-minlen)
      end do

c     scale flux vector magnitudes
      do j=1,nl
        latt(j,3)=latt(j,3)*loglen(j)/len(j)
        latt(j,4)=latt(j,4)*loglen(j)/len(j)
      end do

c     reorder latt array rows for output
c     from 1/2,-1/2,3/2,...,N+1/2,-3/2,...,-N-1/2
c     to N+1/2,...,3/2,1/2,-1/2,-3/2,...,-N-1/2
      do c=1,col-2
c       n=1/2 row
        olat(c+(nrow-1)*(col-2),1)=latt(c,1)
        olat(c+(nrow-1)*(col-2),2)=latt(c,2)
        olat(c+(nrow-1)*(col-2),3)=latt(c,3)
        olat(c+(nrow-1)*(col-2),4)=latt(c,4)
c       n=-1/2 row
        olat(c+nrow*(col-2),1)=latt(c+col-2,1)
        olat(c+nrow*(col-2),2)=latt(c+col-2,2)
        olat(c+nrow*(col-2),3)=latt(c+col-2,3)
        olat(c+nrow*(col-2),4)=latt(c+col-2,4)
c       n>1/2 and n<-1/2 rows
        do r=1,nrow-1
c         n>1/2
          olat(c+(r-1)*(col-2),1)=latt(c+(nrow+1-r)*(col-2),1)
          olat(c+(r-1)*(col-2),2)=latt(c+(nrow+1-r)*(col-2),2)
          olat(c+(r-1)*(col-2),3)=latt(c+(nrow+1-r)*(col-2),3)
          olat(c+(r-1)*(col-2),4)=latt(c+(nrow+1-r)*(col-2),4)
c         n<-1/2
          olat(c+(nrow+r)*(col-2),1)=latt(c+(nrow+r)*(col-2),1)
          olat(c+(nrow+r)*(col-2),2)=latt(c+(nrow+r)*(col-2),2)
          olat(c+(nrow+r)*(col-2),3)=latt(c+(nrow+r)*(col-2),3)
          olat(c+(nrow+r)*(col-2),4)=latt(c+(nrow+r)*(col-2),4)
        end do
      end do

c     output file for energy flux components
      open(unit=30,file="flux.txt",action="write")

c     write atom coordinates, flux vector magnitudes
      do r=1,2*nrow
c       do c=1,col-2-mod(r,5),5
c         j=mod(r,5)+c+(r-1)*(col-2)
        do c=1,col-2
          j=c+(r-1)*(col-2)
          write(30,*) olat(j,1),olat(j,2),olat(j,3),olat(j,4)
        end do
      end do

      close(30)

      end
```

Appendix C

Plotting scripts

C.1 Component plots

The Mathematica script used to make the plots of the horizontal and vertical components from Chapters 3 and 4 is given below.

```
(** Import t, ux, uy lists **)

SetDirectory[];
SetDirectory["Desktop/mode1"];

t = Import["t.txt", "List"];
(*ux=Import["ux.txt","List"];*)
uy = Import["uy.txt", "List"];

Np = Length[t];
Un = uy[[Np + 1]];

(*uxt=Table[{t[[i]],ux[[i]]},{i,1,Np}];*)

uyt = Table[{t[[i]], uy[[i]]}, {i, 1, Np}];

(* plotting options *)
width = 500;
pad = {{65, 10}, {60, 10}};
plot = Directive[Black, Thickness[0.004]];
frame = Directive[Black, Thickness[0.002]];
line = Directive[Black, Thickness[0.003], Dashed];
```

```
(** ux(t) plot **)

(*ListPlot[uxt,Joined->True,PlotRange->{{-2250,2250},{-20,5}
},Frame->True,Axes->False,LabelStyle->(FontFamily
->"CMU Serif"),BaseStyle->{FontSize->20},PlotStyle
->plot,FrameStyle->frame,ImageSize->width,ImagePadding
->pad,FrameLabel->{"t","Subsuperscript[u, 1/2, x](t)"}]*)

(** uy(t) plot **)

tmin = -150;
tmax = 350;

umin = -15;
umax = 65;

ListPlot[uyt, Joined -> True,
 PlotRange -> {{tmin, tmax}, {umin, umax}}, Frame -> True,
 Axes -> False, LabelStyle -> (FontFamily -> "CMU Serif"),
 BaseStyle -> {FontSize -> 20}, PlotStyle -> plot,
 FrameStyle -> frame, ImageSize -> width, ImagePadding -> pad,
 FrameLabel -> {"t", "uy1/2(t)"},
 Epilog -> {line, Line[{{tmin, Un}, {tmax, Un}}]}]
```

C.2 Vector plots

The Mathematica script used to make the vector plots of the lattice energy flux vector from Chapter 5 is given below.

```
(** Import atom coordinates, energy flux vectors from "flux.txt" **)

SetDirectory[];
SetDirectory["Desktop/energy"];

(* vector plot lists *)
```

```
flux = Import["flux.txt", "Table"];
n = Length[flux];
vec = Table[{{flux[[i]][[1]], flux[[i]][[2]]}, {flux[[i]][[3]],
     flux[[i]][[4]]}}, {i, 1, n}];
dots = Table[{flux[[i]][[1]], flux[[i]][[2]]}, {i, 1, n}];
(*mag=Table[{flux[[i]][[1]],flux[[i]][[2]],Sqrt[flux[[i]][[3]]^2+flux[\
[i]][[4]]^2]},{i,1,n}];*)

x = 25;
y = 10;

vec = ListVectorPlot[vec, VectorPoints -> All,
   VectorScale -> {0.006, 6, None},
   VectorColorFunction -> "Rainbow"];
dots = ListPlot[dots, PlotStyle -> Black];
(*mag=ListDensityPlot[mag,ColorFunction->"Rainbow"];*)

Show[vec, dots, PlotRange -> {{-x, x}, {-y, y}}, AspectRatio -> 1/2,
 ImageSize -> Large, LabelStyle -> (FontFamily -> "CMU Serif"),
 BaseStyle -> {FontSize -> 15}, FrameStyle -> Black]
```

Bibliography

[1] D. J. Andrews. Rupture velocity of plane-strain shear cracks. *J. Geophys. Res.*, 81:5679–5687, 1976.

[2] David H. Bailey and Paul N. Swarztrauber. A fast method for the numerical evaluation of continuous Fourier and Laplace transforms. *SIAM Journal on Scientific Computing*, 15(5):1105–1110, 1994.

[3] Chris Behn and M. Marder. The transition from subsonic to supersonic cracks. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 373(2038), 2015.

[4] K. B. Broberg. *Cracks and Fracture.* Academic Press, San Diego, 1999.

[5] Markus J. Buehler, Farid F. Abraham, and Huajian Gao. Hyperelasticity governs dynamic fracture at a critical length scale. *Nature*, 426:141–146, 2003.

[6] R Burridge, G. Conn, and L. B. Freund. The stability of a rapid mode II shear crack with finite cohesive traction. *Journal of Geophysical Research*, 85:2210–2222, 1979.

[7] C. H. Chen, H. P. Zhang, J. Niemczura, K. Ravi-Chandar, and M. Marder. Scaling of crack propagation in rubber sheets. *Europhysics Letters*, 96:36009/1–6, 2011.

[8] R. D. Deegan, P. Petersan, M. Marder, and H. L. Swinney. Oscillating fracture paths in rubber. *Physical Review Letters*, 88:14304, 2002.

[9] J. Fineberg and M. Marder. Instability in dynamic fracture. *Physics Reports*, 313:1–108, 1999.

[10] L. B. Freund. *Dynamic Fracture Mechanics*. Cambridge University Press, Cambridge, 1990.

[11] H Gao, Y Huang, and F F Abraham. Continuum and atomistic studies of intersonic crack propagation. *Journal of the Mechanics and Physics of Solids*, 49:2113–32, 2001.

[12] David J Griffiths. *Introduction to Electrodynamics; 4th ed.* Pearson, Boston, MA, 2013.

[13] T. M. Guozden, E. A. Jagla, and M. Marder. Supersonic cracks in lattice models. *International Journal of Fracture*, page submitted, 2009.

[14] T H Heaton. Evidence for and implications of self-healing pulses of slip in earthquake rupture. *Phys. Earth Planet. Interiors*, 64:1–20, 1990.

[15] Dominic Holland and Michael Marder. Cracks and atoms. *Advanced Materials*, 11:793–806, 1999.

[16] Sh. A. Kulakhmetova, V. A. Saraikin, and L. I. Slepyan. Plane problem of a crack in a lattice. *Mechanics of Solids*, 19:102–108, 1984.

[17] A. Livne, O. Ben-David, and J. Fineberg. Oscillations in rapid fracture. *Physical Review Letters*, 98:124301, 2007.

[18] M Marder. Effect of atoms on brittle fracture. *International Journal of Fracture*, 130:517–555, 2004.

[19] M Marder. Shock-wave theory of rupture of rubber. *Physical Review Letters*, 94:048001/1–4, 2005.

[20] M Marder. Supersonic rupture of rubber. *Journal of the Mechanics and Physics of Solids*, 54:491–532, 2006.

[21] M Marder. Supersonic rupture of rubber. *Journal of the Mechanics and Physics of Solids*, 54:491–532, 2006.

[22] M. Marder and S.P. Gross. Origin of crack tip instabilities. *Journal of the Mechanics and Physics of Solids*, 43:1–48, 1995.

[23] N. I. Muskhelishvili. *Some Basic Problems in the Mathematical Theory of Elasticity.* Noordhoff, Groningen, 1952.

[24] B Noble. *Methods Based on the Wiener-Hopf Technique for the Solution of Partial Differential Equations.* Pergamon, New York, 1958.

[25] P. J. Petersan, R. D. Deegan, M. Marder, and H. L. Swinney. Cracks in rubber under tension exceed the shear wave speed. *Physical Review Letters*, 93:015504/1–4, 2004.

[26] K. Ranjith and J. R. Rice. Slip dynamics at an interface between dissimilar materials. *Journal of the Mechanics and Physics of Solids*, 49:341–361, 2001.

[27] K. Ravi-Chandar. Dynamic Fracture. In I Milne, R O Ritchie, and B Karihaloo, editors, *Comprehensive Structural Integrity Handbook*, volume 2, chapter 5. Elsevier, 2003.

[28] A J Rosakis. Intersonic shear cracks and fault ruptures. *Advances in Physics*, 51:1189–257, 2002.

[29] H. Schardin. Velocity effects in fracture. In B. L. Averbach., editor, *Fracture*, pages 297–330. MIT Press, Cambridge, MA, 1959.

[30] L I Slepyan. *Models and Phenomena in Fracture Mechanics.* Springer, Berlin, 2002.

[31] Leonid Slepyan. Dynamics of a crack in a lattice. *Soviet Physics Doklady*, 26:538–540, 1981.