Theory based machine learning viscoelastic full waveform inversion based on recurrent neural network

Tianze Zhang, Kristopher Innanen, Jian Sun, Daniel Trad

ABSTRACT

In this study, we use a recurrent neural network (RNN) to achieve viscoelastic full waveform inversion. The RNN is a typical type of neural network that consists of several RNN cells. In this study, each RNN cell is designed according to the stress velocity viscoelastic wave equation. With the Automatic Differential engine built in the machine learning library, the exact gradient for the trainable parameters, the velocity models and attenuation models, would be given based on the computational graph. Both the simple and complex model numerical inversion tests prove that the inversion based on this theory-guided recurrent neural network can give accurate inversion results. The performance of this RNN based inversion with different objective functions are also tested. Three objective functions are tested here, which are the l_1 norm, l_2 norm and Huber objective functions. All the three objective functions can provide the right inversion results, however, the l_1 norm and Huber objective function have better accuracy to reconstruct the high wavenumber components of the modes. The l_2 norm inversion has the best data residual convergence rate, but l_1 norm and Huber objective function have better accuracy to reconstruct the models. Compared with Q_p and Q_s, the inversion for V_p, and V_s are more stable with all the three objective functions.

INTRODUCTION

Elasticity is usually considered as a good model for seismic forward and inversion problem. However, in real seismic data, the energy of waves can be converted into different kinds of energy, for instance heat, due to attenuation (Robertsson et al. (1994)). This means that pure elastic media may not be enough to demonstrate the reality in the subsurface in some cases. The quality factor Q is usually used to describe attenuation. The physical meaning of Q is the number of wavelengths a wave must propagate through the material before its amplitude drops by a factor $e^{-\pi}$. By using Pade approximation, Day and Minster (1984) combined the viscoelastic theory into 2D time domain modeling methods. Emmerich and Korn (1987) introduced generalized the standard linear solid (GSLS) model to approximate the viscoelastic earth model. Robertsson et al. (1994) developed the staggered grid finite difference method for viscoelastic modeling Bohlen (2002). The modeling method we use in this study is also based on Robertson's method.

Full waveform inversion (FWI) is a powerful method based on data fitting to invert velocity models. During full waveform inversion, we first generate synthetic wavefields by using initial models. The forward propagating wavefields are calculated, and shotrecords are recorded at the same time to form the synthetic shotrecords. The zero-lag correlation between the backpropagation wavefields and the forward propagation wavefields is calculated as the gradients to update the models. Sometimes, the Hessian matrix would also be calculated to tackle the cross-talk problem. In recent years, adding attenuation into FWI had been studied by many researchers. Fabien-Ouellet et al. (2016) derived the adjoint state

equations for the viscoelastic wave equation based on the velocity-stress formulation, and then performed viscoelastic full waveform inversion on actual fields data. Fabien-Ouellet et al. (2017) explored the use of OpenGL to develop a portable code that can take advantage of the many parallel processor architectures now available, and presented a program for 2D and 3D viscoelastic FWI in the time domain. Yang et al. (2016) studied 3-D multiparameter full waveform inversion (FWI) in viscoelastic media based on the generalized Maxwell/Zener body including an arbitrary number of attenuation mechanisms. Groos et al. (2012) explored which viscoelastic modeling is relevant during a full waveform inversion of shallow seismic surface waves, and concluded that if we use Q factors inaccurate the inversion becomes worse. Trinh et al. (2018) used the SEAM Phase II Foothill dataset to simultaneously invert the P and S wave speed. Belahi et al. (2015) investigated the need to properly account for attenuation when inverting long offset seismic data by comparing the results of elastic FWI applied to viscoelastic data and fully viscoelastic FWI. Belahi et al. (2016) found that inverting for all parameters together is necessary to get access to the short wavelength features of the subsurface model because the short wavelength attenuation model is required to properly treat reflections and converted waves close to the critical angle. According to these researches, the attenuation phenomenon is an important aspect that influences the inversion results of full waveform inversion.

The power of machine learning has become noticed by Geophysicists. One of the most powerful advantages for machine learning is that it can build the linear or nonlinear relationship between the input and output through training trainable parameters. Data-driven methods to invert velocity models have also been studied. Yang and Ma (2019) introduced an inversion method that is based on the convolutional neural networks, which reduces the strong dependency of initial models for scalar wave FWI. Mosser et al. (2018) introduced the generative adversarial networks to perform seismic data inversion. Sun and Demanet (2018) used CNN to expand the bandwidth of the seismic data. Lin and Wu (2018) designed the InverseNet bases on the convolutional network to do full waveform inversion. However, these methods are mainly data-driven, which means that these methods need a huge amount of time for training and ignore the theory we know about inversion and wave propagation. If over-fitting and under-fitting problems occur in these methods, the inversion results would be badly influenced.

Jian et al. (2019) used a recurrent neural network to achieve the scalar wave full waveform inversion. Each of the RNN cells is designed according to the scalar wave equation, which forms a theory-based machine learning seismic data inversion method. In this paper, based on their idea, we introduce the recurrent neural network (RNN) to achieve viscoelastic FWI. In this study, the cells in RNN are designed according to the viscoelastic wave equation. Exact gradients for the trainable parameters are given by the Automatic Differential engine built in the machine learning framework. By using an optimization method and the step length for each model, we can update the models and reduce the misfit between the observed and synthetic data.

VISCOELASTIC WAVE EQUATION

The 2-D first order viscoelastic wave equation can be derived from the momentum conservation equation and the viscoelastic constitutive relationship (for details see appendix).

Equation (1) shows the viscoelastic wave equation written in the stress-velocity form. σ_{xx}, σ_{zz} and σ_{xy} are the stress tensors, v_x and v_y are the velocity fields in horizontal and vertical directions respectively, r_{xx}, r_{yy} and r_{xy} are the memory variables. τ_σ is the relaxation time. It is possible to use the same relaxation time for both of P- and S-waves (Robertsson et al. (1994)). τ_ε^p and τ_ε^s define the attenuation level of the media. π is the relaxation modulus corresponding to P-waves analogous to $\lambda + 2\mu$ in the elastic case, where λ and μ are the Lame constants. μ is the relaxation modulus corresponding to S- waves and is the analog of the Lame constant μ in the elastic case. Flowing Robertsson's step (Robertsson et al. (1994)), we have the viscoelastic wave equation:

$$
\begin{cases}
\dfrac{\partial v_x}{\partial t} = \dfrac{1}{\rho}\left(\dfrac{\partial \sigma_{xx}}{\partial x} + \dfrac{\partial \sigma_{xy}}{\partial y}\right) \\
\dfrac{\partial v_y}{\partial t} = \dfrac{1}{\rho}\left(\dfrac{\partial \sigma_{xy}}{\partial x} + \dfrac{\partial \sigma_{yy}}{\partial y}\right) \\
\dfrac{\partial \sigma_{xy}}{\partial t} = \mu\dfrac{\tau_\varepsilon^s}{\tau_\sigma}\left(\dfrac{\partial v_x}{\partial y} + \dfrac{\partial v_y}{\partial x}\right) + r_{xy} \\
\dfrac{\partial \sigma_{xx}}{\partial t} = \pi\dfrac{\tau_\varepsilon^p}{\tau_\sigma}\left(\dfrac{\partial v_x}{\partial x} + \dfrac{\partial v_y}{\partial y}\right) - 2\mu\dfrac{\tau_\varepsilon^s}{\tau_\sigma}\dfrac{\partial v_y}{\partial y} + r_{xx} \\
\dfrac{\partial \sigma_{yy}}{\partial t} = \pi\dfrac{\tau_\varepsilon^p}{\tau_\sigma}\left(\dfrac{\partial v_x}{\partial x} + \dfrac{\partial v_y}{\partial y}\right) - 2\mu\dfrac{\tau_\varepsilon^s}{\tau_\sigma}\dfrac{\partial v_x}{\partial x} + r_{yy} \\
\dfrac{\partial r_{xy}}{\partial t} = -\dfrac{1}{\tau_\sigma}\left(r_{xy} + \mu\left(\dfrac{\tau_\varepsilon^s}{\tau_\sigma} - 1\right)\left(\dfrac{\partial v_x}{\partial y} + \dfrac{\partial v_y}{\partial x}\right)\right) \\
\dfrac{\partial r_{xx}}{\partial t} = -\dfrac{1}{\tau_\sigma}\left(r_{xx} + \pi\left(\dfrac{\tau_\varepsilon^p}{\tau_\sigma} - 1\right)\left(\dfrac{\partial v_x}{\partial x} + \dfrac{\partial v_y}{\partial y}\right) - 2\mu\left(\dfrac{\tau_\varepsilon^s}{\tau_\sigma} - 1\right)\dfrac{\partial v_y}{\partial y}\right) \\
\dfrac{\partial r_{yy}}{\partial t} = -\dfrac{1}{\tau_\sigma}\left(r_{yy} + \pi\left(\dfrac{\tau_\varepsilon^p}{\tau_\sigma} - 1\right)\left(\dfrac{\partial v_x}{\partial x} + \dfrac{\partial v_y}{\partial y}\right) - 2\mu\left(\dfrac{\tau_\varepsilon^s}{\tau_\sigma} - 1\right)\dfrac{\partial v_x}{\partial x}\right)
\end{cases}
\tag{1}
$$

Equation (1) shows the viscoelastic wave equation we use in this study. The partial derivatives in time and space are approximated by second order finite-difference on a staggered grid, in which velocities are updated at integer time steps Δt, the stress fields and memory variables are updated at half-time steps. In this paper, the particle derivative is calculated according to the image convolution. We first create kernels to calculate partial derivative in different directions according to the staggered grid method. The kernels would scan the wavefields and calculate image convolution to get the partial derivatives.

RECURRENT NEURAL NETWORK

The recurrent neural network (RNN) is a typical type of network in deep learning, which is suitable to deal with data that be discrete in time. The RNN usually consists of a series of cells, each represents in time step. In each RNN cell, there are one or more trainable parameters waiting to be updated. According to our designed structure, each cell uses the input data to generate output data. Meanwhile, a Dynamic Computational Graph would also form, which records every mathematical operation between the variables and trainable parameters. The misfit between the output data generated by RNN and the labeled data would be calculated according to the misfit function we set up. The partial derivative

of the residual with respect to the trainable parameters would be calculated according to the backpropagation of the Dynamic Computational Graph.

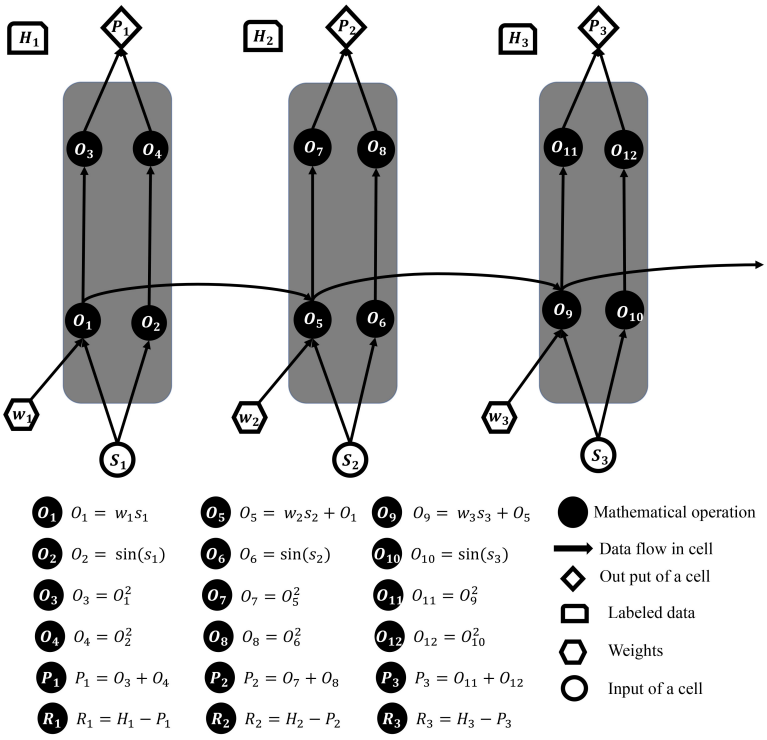

FIG. 1. Forward propagation for RNN

Figure 1 shows a simple RNN network, which represents a mathematical relationship between the input data sequence **S** and output data sequence **P**. Figure 1 shows three RNN cells. In each cell, the black circles are mathematical operations. The hollow circles are the input data. The hollow rhombus are the outputs of the cells. The hollow hexagons are the trainable parameters. The sequence **H** is the labeled data at each time step. We want to build the right relationship between the input sequence **S** and output sequence **P** by optimizing the trainable parameters. The sequence $\mathbf{O} = O_1, O_2, ...$ represents internal variables. In Figure 1, we can see that O_1 is a variable generated in cell number one and used by cell number two. O_5 is the variable generated in cell number two and used by cell number three. The data generated by one cell is used recurrently by the next cell. This is why it is called a recurrent neural network. According to the mathematical operations in the RNN cell demonstrated under Figure 1, we can get the final outputs of the RNN at each time, which forms the sequence **P**. Every variable in the cell is a scalar.

After we finish the forward propagation, we need to update the weights in RNN. The gradients are calculated by using the backpropagation method according to the Chain's Rule. Figure 2 shows the backword propagation for this RNN. Take $t = 1$ as an example. The residual R_1 is calculated with $R_1 = H_1 - P_1$. Sequence $\mathbf{R} = R_1, R_2, ...$ is evaluating the difference between the output of the cell and the labeled data. If we need to update the trainable parameter w_1, then we need to calculate the partial derivative of the residual with

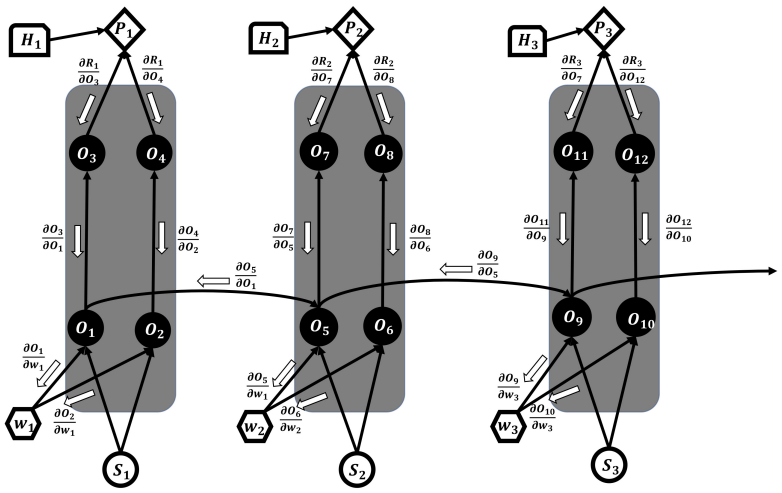

FIG. 2. Backword propagation for RNN

respect to the trainable parameter w_1 with the backpropagation method:

$$\frac{\partial R_1}{\partial w_1} = \frac{\partial R_1}{\partial P_1}\frac{\partial P_1}{\partial O_3}\frac{\partial O_3}{\partial w_1} = \frac{\partial R_1}{\partial P_1}\frac{\partial P_1}{\partial O_3}\frac{\partial O_3}{\partial O_1}\frac{\partial O_1}{\partial w_1} = -2O_1 s_1 = -2w_1 s_1^2 \tag{2}$$

This partial derivative is the gradient for w_1 at time step one. Following the same process, at time step two, we can also calculate the partial derivative of the residual with respect to weight w_1:

$$\frac{\partial R_2}{\partial w_1} = \frac{\partial R_2}{\partial P_2}\frac{\partial P_2}{\partial w_1} = \frac{\partial R_2}{\partial P_2}\frac{\partial P_2}{\partial O_7}\frac{\partial O_7}{\partial w_1} = \frac{\partial R_2}{\partial P_2}\frac{\partial P_2}{\partial O_7}\frac{\partial O_7}{\partial O_5}\frac{\partial O_5}{\partial O_1}\frac{\partial O_1}{\partial w_1} = -2O_5 s_1 \tag{3}$$

Similarly, we have:

$$\frac{\partial R_3}{\partial w_1} = \frac{\partial R_3}{\partial P_3}\frac{\partial P_3}{\partial w_1} = \frac{\partial R_3}{\partial P_3}\frac{\partial P_3}{\partial O_{11}}\frac{\partial O_{11}}{\partial w_1} = \frac{\partial R_3}{\partial P_3}\frac{\partial P_3}{\partial O_{11}}\frac{\partial O_{11}}{\partial O_9}\frac{\partial O_9}{\partial O_5}\frac{\partial O_5}{\partial O_1}\frac{\partial O_1}{\partial w_1} = -2O_9 s_1 \tag{4}$$

Thus, if the RNN has propagated three times, the gradient of w_1, which is also the partial derivative of the loss with respect to w_1 can be expressed as:

$$\frac{\partial Loss}{\partial w_1} = -2s_1(O_1 + O_5 + O_9) \tag{5}$$

The gradients for other parameters can also be calculated by using the same principle.

We now give the basic steps required train a RNN.

(a) Design the RNN cells by considering which kind of mathematical relationship between the input data and output data should have. Set up which parameter should be considered as the trainable parameter.

(b) Calculate the forward propagation by using the input data.

(c) Calculate the residual between the output data generated by RNN and labeled data.

(d) By using the backpropagation method, or the Chain's Rule, get the gradients for the trainable parameters.

(e) Implement an optimization method to calculate the direction and use the step length to update the trainable parameters.

(f) Start another iteration.

By using a machine learning library, like Tensor Flow or Pytorch, we do not need to calculate the gradients all by ourselves. The gradients can be calculated by using the Automatic Differential engine built-in these machine learning frameworks. During the forward propagation, every mathematical operation is recorded in the Dynamic Computational Graph. The gradients are calculated according to this Dynamic Computational Graph by using the backpropagation method explained above. Also, the gradients calculated in this way are the exact gradients based on your computational graph.

The recurrent neural network we design in this study represents elastic wave equation we discussed in the first section. That is why it is a theory-based machine learning method. Figure 3 shows the structure of the viscoelastic RNN cell we build in this study. The green circles in Figure 3 represent the mathematical operations. The light blue oval means the stress fields. A purple oval means velocity field and the gray ovals are the memory variables. The yellow boxes are the trainable parameters. The cross line with a black dot on it means that several data are running parallel in the cell. The cross line without a black dot means that it is simple overlapping. The brown line means we need to send data out of the cell. The light blue line means that we need to send data into the cell.

Figure 4 shows how the viscoelastic RNN generates synthetic data. As we can see from figure 4, the network consists of several viscoelastic RNN cells. Each RNN cell is designed according to the viscoelastic wave equation, and considered as a time step. At each time step, the input is the discrete source wavelet information, and the corresponding stress fields, velocity fields and memory variables at that time would be calculated in that cell. Figure 4 only shows the velocity field in x direction, however, the stress fields, the velocity fields, and the memory variables would all be calculated in this cell. After we have the fields, the receivers on the top of the model would record the waveform information at that time, which forms the synthetic shotrecords. Then, the wavefields calculated at this time would be sent to the next cell to calculate the wavefields for the next time. At the end of the computational time, we will have the final shotrecords.

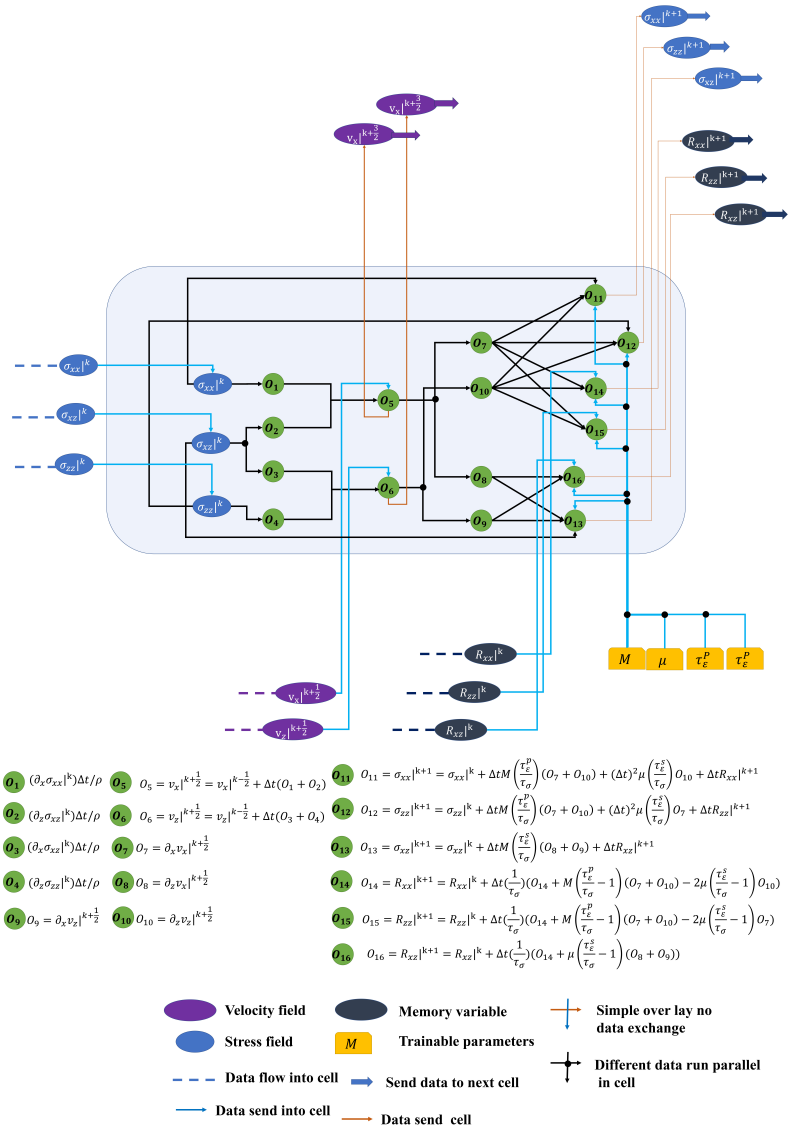

FIG. 3. Viscoelastic RNN cell

NUMERICAL TESTING

In this section, we will perform some numerical tests to examine the efficiency of the methods proposed. Figure 5 shows the shotrecords of one trace generated by the viscoelastic RNN with different levels of attenuation and reference frequency. From the shotrecords we can see that with the increase of the attenuation level, the amplitude has decreases, which means that with the increase of the attenuation level, the energy of the wave is being lost and transformed into other kinds of energy, for instance, heat. Also, with different reference frequencies, the phase of the shotrecords also shifts in different directions. According to figure 5, with a higher reference frequency, as the increase of the attenuation level, the peak of the records shifts to the left. However, with a lower reference frequency, as the increase of the attenuation level, the peak of the records does not shift much. This means that both the attenuation level and the chosen reference frequency influence the modeling of the records, and therefore influence the inversion results.

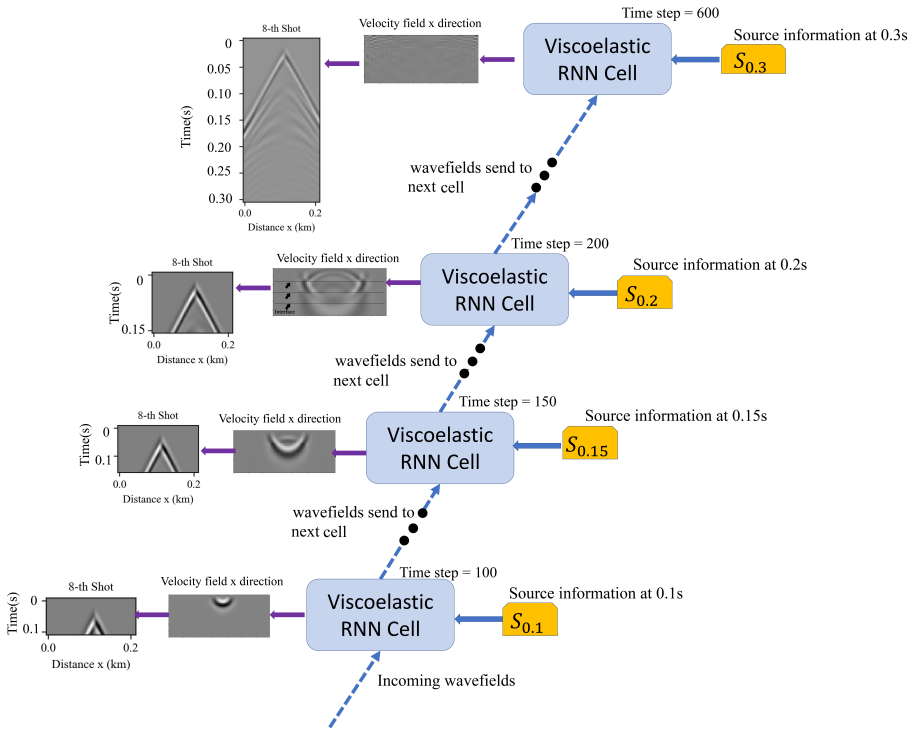

FIG. 4. How viscoelastic RNN generate synthetic data

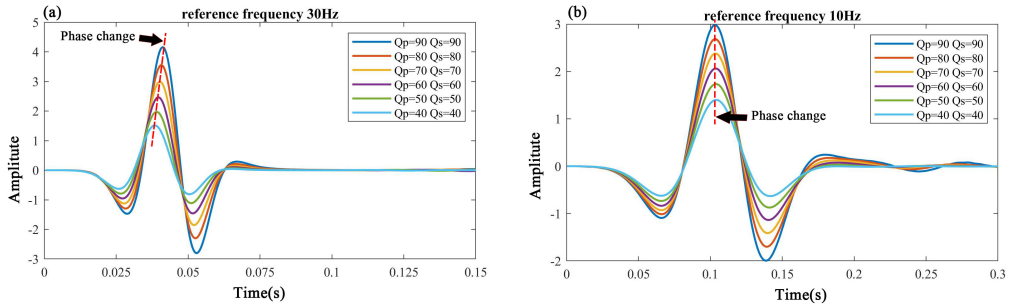

FIG. 5. Amplitude and phase change due to different attenuation levels and reference frequencies

Now, we will test the inversion method by using synthetic data. Figure 6 demonstrates the true V_p, V_s, Q_p and Q_s model. The model consists of four layers, and each layer has its own P wave velocity, S wave velocity and attenuation levels. The size of the model is 30×70 grid points. The source is the Ricker's wavelet with the a frequency of $35Hz$. The shots are evenly distributed on the top of the model at every 5 grid points. The receivers are located on every grid point of the model. The grid lengths in horizontal and vertical directions are $dx = 3m$ and $dz = 3m$ respectively. The total receiving time is $0.2s$, and the time step is $0.0005s$. Figure 7 shows the first 4 shotrecords in horizontal direction and vertical direction respectively generated by viscoelastic RNN. These shotrecords are generated with the true model and we use these data as the labeled data. In this test, the initial V_p, V_s, Q_p and Q_s models will be used0 as the trainable parameters, and the initial models are obtained by smoothing the true models. During the forward propagation, as the synthetic records are calculated, they are recorded to build the Dynamic Computational Graph. The gradients of the trainable parameters would be calculated by using this Graph, according to the backpropagation method. After, the gradients are calculated, with an

optimization method and the step lengths for each model, we can get the directions to get the trainable parameter updates.

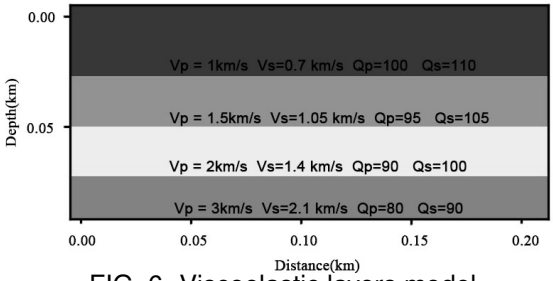

FIG. 6. Viscoelastic layers model

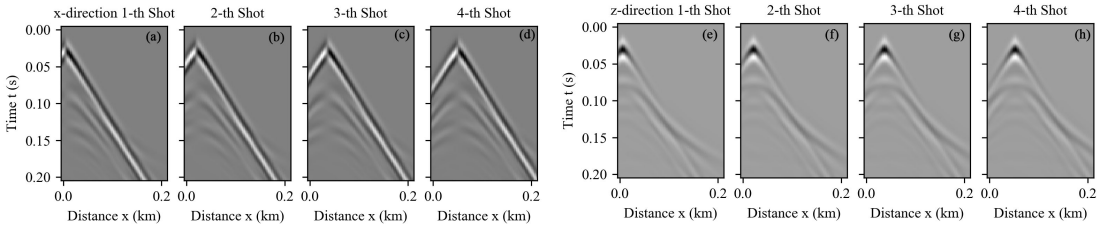

FIG. 7. Viscoelastic shotrecords generated by viscoelastic RNN at different source locations

Figures 8 (b), (e), (h) and (k) are the initial models for V_p, V_s, Q_p and Q_s respectively, and figure 8 (c), (f), (i), (l) are the inversion results based on recurrent neural network full waveform inversion. From the comparison of the true model and the inversion results, we can see that FWI based on viscoelastic RNN is promising. The main layers of the models have been correctly updated. Figure 9 shows the comparison of the inversion result at $0.1km$ for V_p, V_s Q_p and Q_s respectively. From the inversion results, we can also see that both the value and position of the velocity layers have been correctly recovered.

Figure 10 shows the inversion results for another model with a more complex structure. The size of the model is 70×100, and the grid lengths in x and z directions are $dx = dz = 2.5m$. 7 sources are evenly distributed on the surface of the model. We also use the Ricker's wavelet as the sources. The main frequency of the source is $35Hz$. The time step is $0.005s$, and the total receiving time for the shotrecords is $0.3s$, which means that there are 600 time steps in this RNN network. Figure 10 (a), (d), (g) and (j) are the true models for V_p, V_s, Q_p and Q_s model. Figure 10 (b), (e), (h) and (k) are the initial models for V_p, V_s, Q_p and Q_s model, and figure 10 (c), (f), (i) and (l) are the inversion results. We can see that the structure of the model has also been accurately reconstructed. The agreement with the true model is very satisfactory, and the lower parts of the models have also been correctly updated. This inversion took about 3 hours and 4 parameters were simultaneously updated on a Mac, with 16Gb of RAM and an i9-9880H CPU (2.30GHz).

INVERSION WITH DIFFERENT OBJECTIVE FUNCTION

Full waveform inversion is an ill-posed data fitting inversion problem. Initial models, the definition of the multiparameter classes and inaccurate modeling of wavefield amplitudes would all influence the inversion results. More robust objective functions have been introduced by researchers to increase the stability for FWI (Brossier et al. (2010)). In this section, we will investigate how different objective functions would influence the inversion

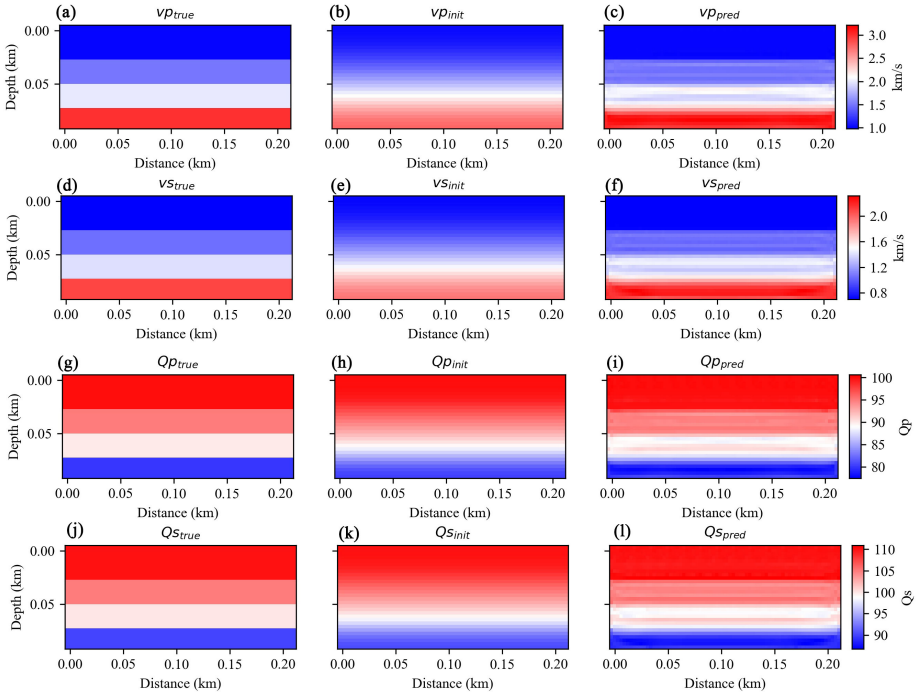

FIG. 8. (a),(b),(c): True model, initial model and inversion result for P wave velocity model respectively, (d),(e),(f): True model, initial model and inversion result for S wave velocity model respectively, (g),(h),(i): True model, initial model and inversion result for Q_p model respectively, (j),(k),(l): True model, initial model and inversion result for Q_s model respectively

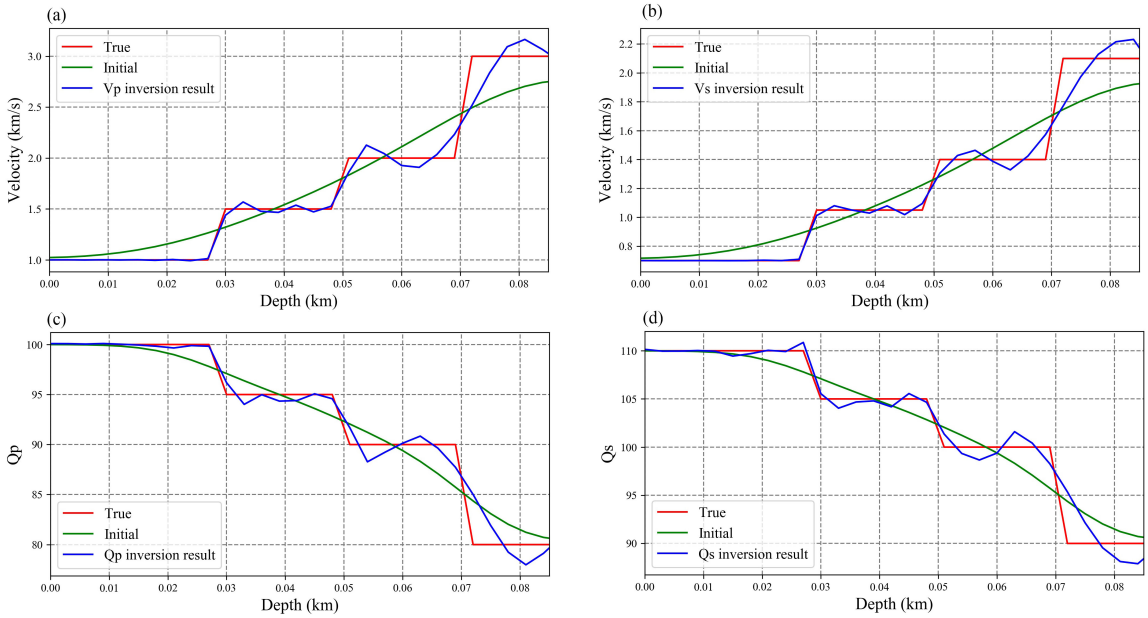

FIG. 9. (a) V_p inversion result at 0.1km, (b) V_s inversion result at 0.1km (c) Q_p inversion result at 0.1km (d) Q_s inversion result at 0.1km

results with the viscoelastic RNN we proposed. For traditional FWI, when we change the objective function, we need to derive the gradients again. However, if the inversion is under the framework of a recurrent neural network, this process can be done by simply change the objective function. The gradients based on that objective function would be calculated automatically by using the Automatic Differential engine in machine learning library we use.

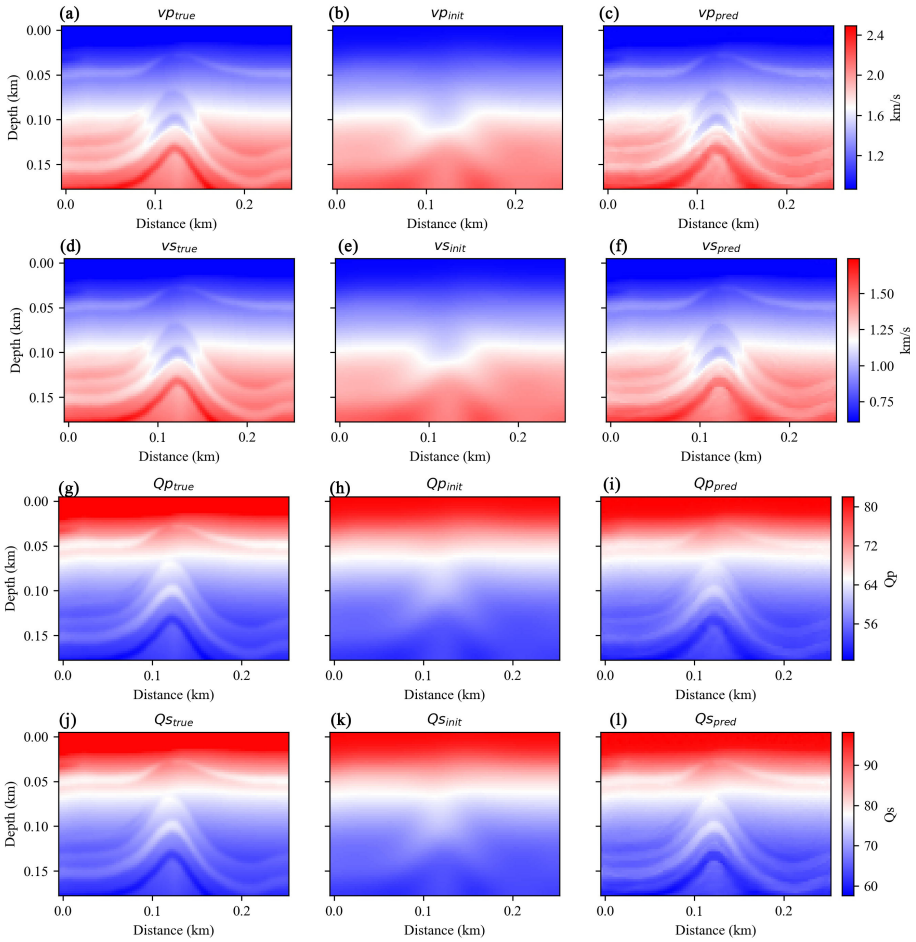

FIG. 10. (a), (d), (g) and (j) true model for V_p, V_s, Q_p and Q_s, (b), (e), (h) and (k) initial model for V_p, V_s, Q_p and Q_s, (c), (f), (i) and (l) inversion results for V_p, V_s, Q_p and Q_s,

This is also why the RNN is a powerful inversion tool. The Automatic Differential engine relieves us from the burden of deriving gradients, and these are the exact gradients based on the computational graph. Thus, we can focus on other aspects of the seismic inversion problem, for instance, the modeling error problems or more robust objective functions.

In this section, three objective functions would be tested, which are the l_1 norm function, the l_2 norm function, and the Huber function.

The loss function for l_1 is:

$$loss_{l_1} = |x - y| \tag{6}$$

The loss function for l_2 is:

$$loss_{l_2} = \frac{1}{2}(x - y)^2 \tag{7}$$

The loss function for Huber function is:

$$loss_{Huber} = \begin{cases} 0.5(x-y)^2, if|x-y| < 1 \\ |x-y| - 0.5, otherwise \end{cases} \tag{8}$$

From the definition for the loss functions, we can see that l_2 objective function is the least-squares norm, and l_1 objective function is the least-absolute values norm. The Huber objective function is the combination of the l_1 norm and the l_2 norm. The l_2 norm is the misfit we use in the traditional FWI. Djikpéssé and Tarantola (1999) successfully used the l_1 norm objective function to invert field data from the Gulf of Mexico with time-domain FWI. Pyun et al. (2009) also studied the l_1 norm performance on FWI in frequency domain. Their experiments demonstrate that the inversion based on l_1 is more robust than l_2 inversion. In this section, we will test the performance of these objective functions on the viscoelastic FWI under the framework of the recurrent neural network.

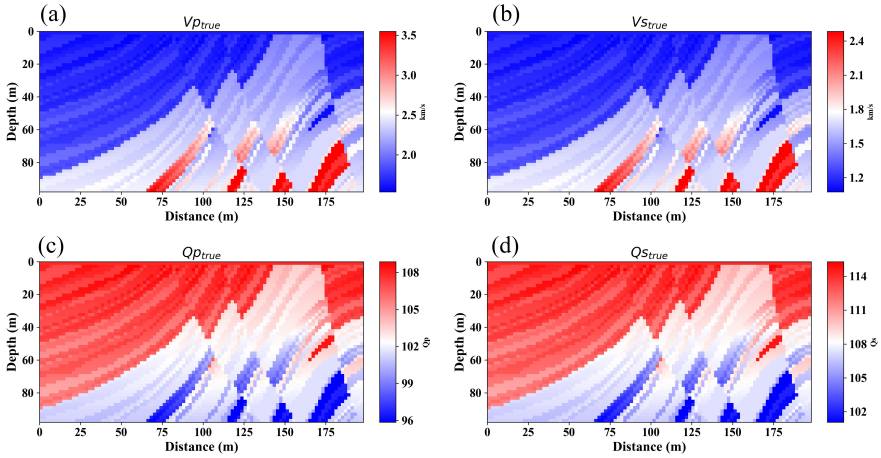

FIG. 11. (a), True Vp model, (b) True Vs model, (c) True Qp model, (d) True Qs model

In this numerical test, we use part of the Marmousi model to test the efficiency of this inversion method. The source wavelet is the Ricker's wavelet with a main frequency of 35Hz. The model size is 50×100 grid points. The grid length of the model is $dz = dx = 2m$. 7 shots are evenly distributed on the top of the model. Every grid point at the top of the model contains a receiver. The total receiving time for the shotrecords is $0.3s$, and the time step is $0.004s$. We use the 2 order in time and 2 order in space staggered grid method to simulate the synthetic data. Figure 11 (a), (b), (c) and (d) are the true models for V_p, V_s, Q_p and Q_s. Figure 12 (a), (e), (i), (m) are the initial models for V_p, V_s, Q_p and Q_s respectively.

Figure 12 (d), (h), (i),(l) and (p) are the inversion results by using the l_2 norm objective function for V_p. V_s, Q_p and Q_s. Figure 12 (c), (g), (k), and (o) are the inversion results by using the l_1 norm objective function for V_p. V_s, Q_p and Q_s. Figure 12 (b), (f), (j) and (n) are the inversion results for the Huber misfit function for V_p. V_s, Q_p and Q_s model respectively. From the inversion results we can see that by using all the three objective functions, the main structures of the models have been correctly updated. It is fair to say that all the inversions can converge to the right solution for this inversion problem. This could also be proven by the profile plot in Figure 13, which plots the initial value, true value and inversion result at different locations of the models. The plots in figure 13 demonstrate that the inversion results are correctly updated and get close to the true value.

However, in the inversion results for V_p and V_s, the very high wavenumber components of the model are better reconstructed by using the Huber and l_1 objective function. On

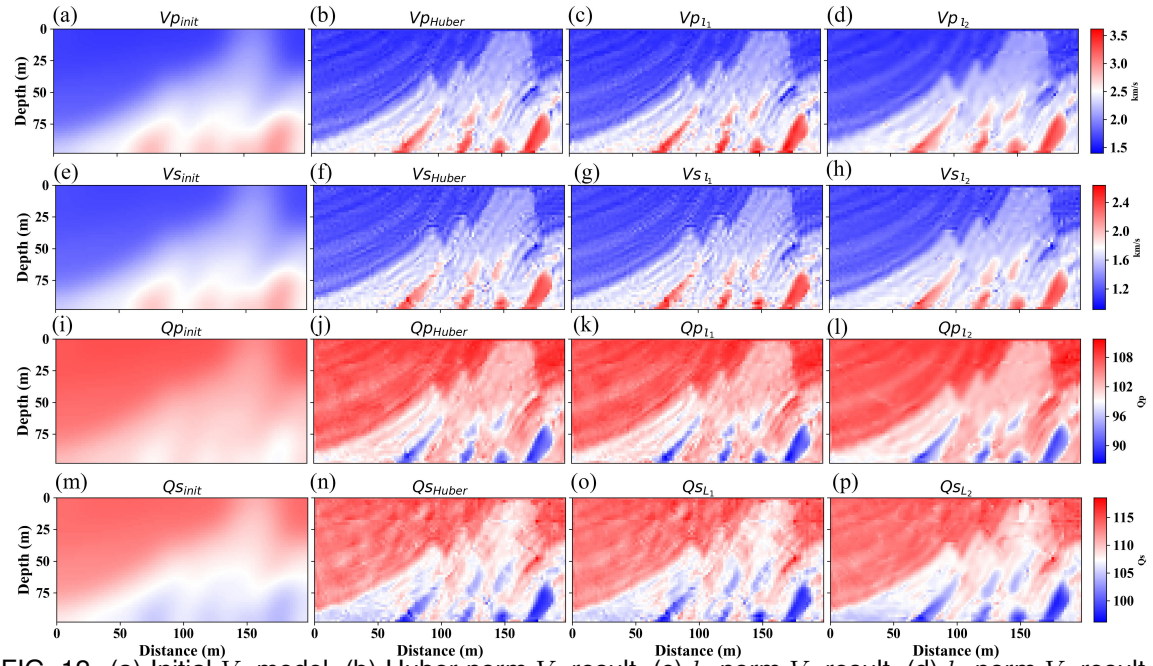

FIG. 12. (a) Initial V_p model, (b) Huber norm V_p result, (c) l_1 norm V_p result, (d) l_2 norm V_p result, (e) Initial V_s model, (f) Huber norm V_s result, (g) l_1 norm V_s result, (h) l_2 norm V_s result, (i) Initial Q_p model, (j) Huber norm Q_p result, (k) l_1 norm Q_p result, (l) l_2 norm Q_p result, (m) Initial Q_s model, (n) Huber norm Q_s result, (o) l_1 norm Q_s result, (p) l_2 norm Q_s result

the one hand, the bandwidth-limited problem may cause this inaccuracy problem for the very high component of the models. On the other hand, this may also due to the nature of using different objective functions to perform inversion, which means that the l_1 and Huber function would be more robust objective functions for the full waveform inversion. From figure 12 we can also see that the high attenuation part of the model has been correctly updated. However, compared with V_p, and V_s, the very high wavenumber of the inversion results for Q_p and Q_s is not very well constructed. This may due to the cross-talk problem. Cross talk problem appears when we need to simultaneously update the models. From the inversion results we can see that for all of the four models, the V_s model suffers the most from the cross-talk problem. In figure 13 (g),(h) and (i), we can see that the inversion result for Q_p by using the l_2 norm is less accurate compared with the inversion results with l_1 and Huber misfit function, as the black arrows pointing out.

Figure 14 shows how normalized data residual declines as the increase of the iteration times. The solid line represents the l_1 norm data residual. The dotted line is the Huber norm data residual, and the solid dotted line stands for the l_2 norm data residual. We can see that the l_2 norm has the fastest convergence rate compared with the other two objective functions at the early periods of the iteration times. Figure 15 shows how the model misfit changes as the increase of iteration times. Figure 15 (a) is the V_p model misfit. Figure 15 (b) is the V_s model misfit. Figure 15 (c) is the Q_p model misfit. Figure 15 (d) is the Q_s model misfit. The solid lines are the misfits for l_1 norm. The dotted lines are the misfits for Huber norm. The solid dotted lines are the misfits for l_2 norm. In these figures we can see that the Huber norm can provide more accurate models for V_p, V_s and Q_p. While for Q_s, the l_1 norm has the best model misfit convergence rate. The model data misfits declination for V_p and V_s are very stable, however for Q_p and Q_s, the convergence rates are not very

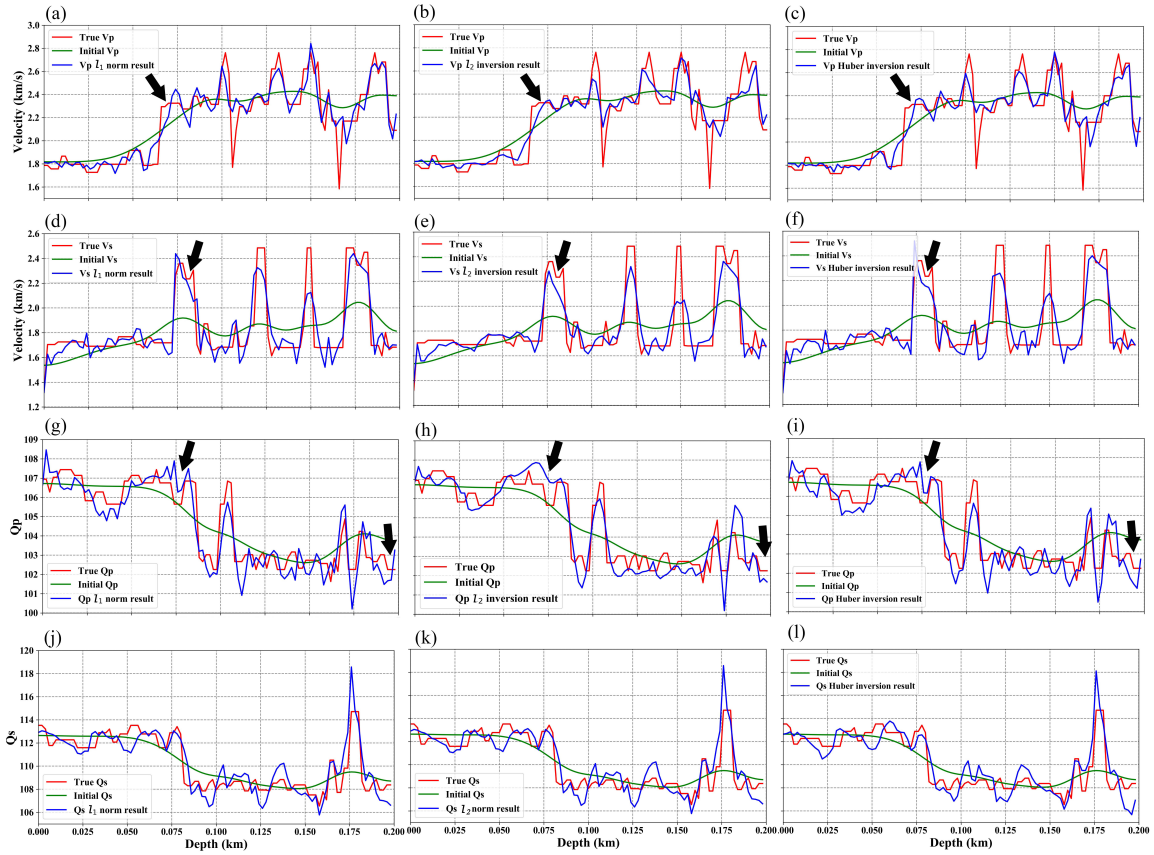

FIG. 13. (a) V_p l_1 norm inversion result at 60m, (b) V_p l_2 norm inversion result at 60m, (c) V_p Huber norm inversion result at 60m, (d) V_s l_1 norm inversion result at 90m, (e) V_s l_2 norm inversion result at 90m, (f) V_s Huber norm inversion result at 90m, (g) Q_p l_1 norm inversion result at 42m, (h) Q_p l_2 norm inversion result at 42m, (i) Q_p Huber norm inversion result at 42m, (j) Q_s l_1 norm inversion result at 48m, (k) Q_s l_2 norm inversion result at 48m, (l) Q_s Huber norm inversion result at 48m,

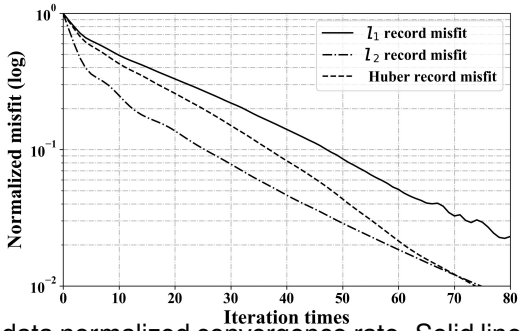

FIG. 14. Seismic records data normalized convergence rate. Solid line: l_1 norm. Dotted line: Huber norm.Solid dotted line: l_2 norm.

stable, especially for Q_s. All the misfits for Q_p and Q_s vibrate several times before they get converged. From these figures, we can see that, in general, l_2 norm has a better convergence rate for the seismic data records, however, the l_1 and Huber objective function could more accurately reconstruct the models.

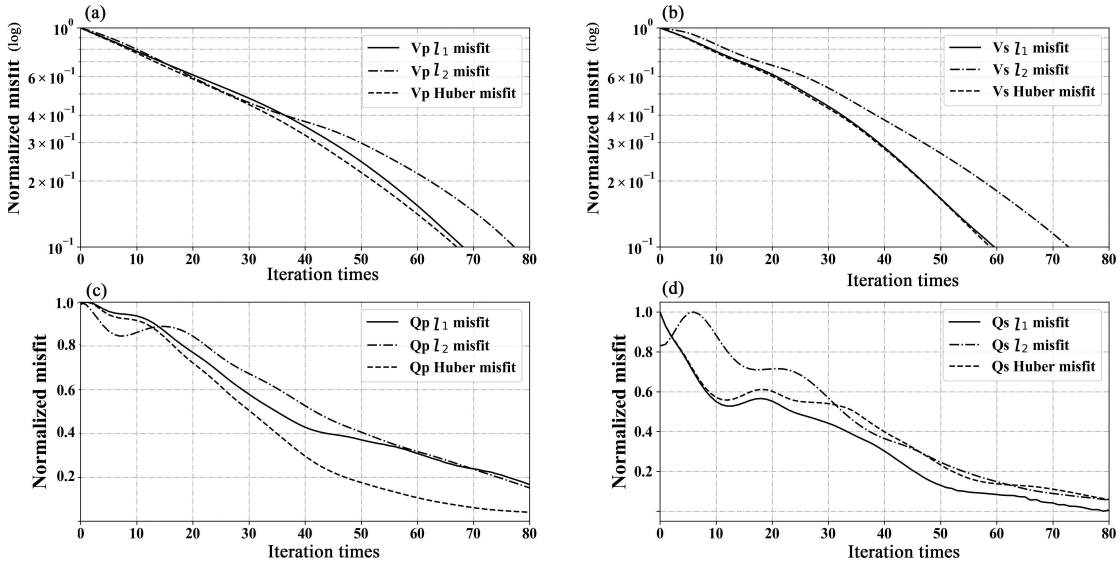

FIG. 15. Model misfit

ACKNOWLEDGMENT

We thank the sponsors of CREWES for continued support. This work was funded by CREWES industrial sponsors, and NSERC (Natural Science and Engineering Research Council of Canada) through the grant CRDPJ 461179-13. We also thanks the support from the China Scholarship Council (CSC).

CONCLUSION

In this study, we first give a brief introduction to the recurrent neural network and demonstrate how the trainable parameters are updated in the RNN. And then, based on the viscoelastic wave equation, we build the viscoelastic RNN cell and performed the viscoelastic full waveform inversion, which forms the theory-based machine learning full waveform inversion. This inversion framework generates the gradient automatically, and the gradients are the exact gradients based on the forward computational graph. Numerical tests show that the inversion framework we introduced generates the correct inversion results for V_p, V_s, Q_p and Q_s. We also test the performance of different misfits for viscoelastic full waveform inversion based on this RNN framework. Three objective functions are tested here,which are the l_1, l_2, and Huber norm functions. All, the three objective functions have given us the right inversion results, however, the l_1 and Huber norm have the better ability to reconstruct the high wavenumber component of the models. From the data residual declination rate, we can see that the l_2 misfit has the fastest convergence rate. The inversion for V_p and V_s are more stable than the inversion for Q_p and Q_s. All three model objective functions for Q_p and Q_s vibrate several times during the inversion process.

APPENDIX

The constitutive relationship for a vicoelastic media can be expressed as:

$$\sigma_{ij} = G_{ijkl} * \dot{\epsilon}_{kl} = \dot{G}_{ijkl} * \epsilon_{kl}, \tag{9}$$

, where the G is a tensor called the relaxation function. $*$ means the time convolution. The dot means the time differential. σ_{ij} means the stress and ϵ_{kl} demonstrate the strain. In a GSLS framework, (Liu et al. (1976)), the relaxation function could be described as :

$$G(t) = M_R \left(1 - \sum_{l=1}^{L} \left(1 - \frac{\tau_{\varepsilon l}}{\tau_{\sigma l}}\right) e^{-t/\tau_{\sigma l}}\right) \theta(t), \tag{10}$$

, where M_R is the relaxed modulus, L is the number of layers for viscoelastic model $\tau_{\sigma l}$ and $\tau_{\varepsilon l}$ are the relaxation time for stress and strain. When it comes to isotropic viscoelastic media. equation (1) becomes:

$$\sigma_{ij} = \dot{\Lambda} * \delta_{ij}\epsilon_{kk} + 2\dot{M} * \epsilon_{ij}, \tag{11}$$

Einstein's notation is used in this formula and the δ_{ij} is the Kronecker delta. Λ and M are the relaxation functions respectively. Here we define:

$$\Pi = \Lambda + M, \tag{12}$$

and according to the relaxation function based on GSLS framework we have:

$$\Pi = \pi \left(1 - \sum_{l=1}^{L} \left(1 - \frac{\tau_{el}^{P}}{\tau_{\sigma l}^{P}}\right) e^{-t/\tau_{\sigma l}^{P}}\right) \theta(t), \tag{13}$$

,where $\tau_{\varepsilon l}^{P}$, and $\tau_{\sigma l}^{P}$ are the strain and stress relaxation time for P wave. $\theta(t)$ is the Heaviside function, $\pi = \lambda + 2\mu$ and λ and μ are the elastic Lame parameters. Similarly, we have:

$$M = \mu \left(1 - \sum_{l=1}^{L} \left(1 - \frac{\tau_{el}^{S}}{\tau_{\sigma l}^{S}}\right) e^{-t/\tau_{\sigma l}^{S}}\right) \theta(t), \tag{14}$$

, where $\tau_{\varepsilon l}^{S}$, and $\tau_{\sigma l}^{S}$ are the strain and stress relaxation time for S wave. According to the relationship between the velocity field, the stress and strain we have :

$$\dot{\epsilon_{ij}} = \frac{1}{2}(\partial_i v_j + \partial_j v_i), \tag{15}$$

$$\dot{\sigma}_{ii} = (\dot{\Pi} - 2\dot{M}) * \partial_k v_k + 2\dot{M} * \partial_i v_i, \tag{16}$$

$$\dot{\sigma}_{ij} = \dot{M} * (\partial_i v_j + \partial_j v_i), \tag{17}$$

, the time differential of Π is :

$$\dot{\Pi} = \pi \left(\frac{1}{\tau_{ol}} \sum_{\ell=1}^{L} \left(1 - \frac{\tau_{e\ell}^{P}}{\tau_{\sigma\ell}^{P}}\right) \mathrm{e}^{-t/\tau_{\sigma l}^{P}}\right) \theta(t) + \pi \left(1 - \sum_{\ell=1}^{L} \left(1 - \frac{\tau_{e\ell}^{P}}{\tau_{\sigma\ell}^{P}}\right) \mathrm{e}^{-t/\tau_{\sigma l}^{P}}\right) \delta(t), \tag{18}$$

Similarly, we have :

$$\dot{M} = \mu \left(\frac{1}{\tau_{ol}} \sum_{\ell=1}^{L} \left(1 - \frac{\tau_{e\ell}^{S}}{\tau_{\sigma\ell}^{S}} \right) \mathrm{e}^{-t/\tau_{\sigma l}^{S}} \right) \theta(t) + \mu \left(1 - \sum_{\ell=1}^{L} \left(1 - \frac{\tau_{e\ell}^{S}}{\tau_{\sigma\ell}^{S}} \right) \mathrm{e}^{-t/\tau_{\sigma l}^{S}} \right) \delta(t), \quad (19)$$

Now, we subscribe equation (11) and equation (12) into equation (8) and (9), and change i, j, into x and z we have :

$$\begin{aligned}
\dot{\sigma}_{xx} =& (\dot{\Pi} - 2\dot{M}) * (\partial_x v_x + \partial_z v_z) + 2\dot{M} * \partial_x v_x \\
=& \left\{ \pi \left[1 - \sum_{l=1}^{L} \left(1 - \frac{\tau_{\varepsilon l}^{P}}{\tau_{\sigma l}^{P}} \right) \right] - 2\mu \left[1 - \sum_{l=1}^{L} \left(1 - \frac{\tau_{\varepsilon l}^{S}}{\tau_{\sigma l}^{S}} \right) \right] \right\} \\
& \times \left(\frac{\partial v_x}{\partial x} + \frac{\partial v_z}{\partial z} \right) + 2\mu \left[1 - \sum_{l=1}^{L} \left(1 - \frac{\tau_{\varepsilon l}^{S}}{\tau_{\sigma l}^{S}} \right) \right] \frac{\partial v_x}{\partial x} + \\
& \sum_{l=1}^{L} \pi \frac{1}{\tau_{\sigma l}^{P}} \left(1 - \frac{\tau_{el}^{P}}{\tau_{\sigma l}^{P}} \right) e^{-t/\tau_{\sigma l}^{P}} \theta(t) * \left(\frac{\partial v_x}{\partial x} + \frac{\partial v_z}{\partial z} \right) + \\
& \sum_{l=1}^{L} -2\mu \frac{1}{\tau_{\sigma l}^{S}} \left(1 - \frac{\tau_{\varepsilon l}^{S}}{\tau_{\sigma l}^{S}} \right) e^{-t/r_{\sigma l}^{S}} \theta(t) * \frac{\partial v_z}{\partial z},
\end{aligned} \quad (20)$$

$$\begin{aligned}
\dot{\sigma}_{zz} =& (\dot{\Pi} - 2\dot{M}) * (\partial_x v_x + \partial_z v_z) + 2\dot{M} * \partial_z v_z \\
=& \left\{ \pi \left[1 - \sum_{l=1}^{L} \left(1 - \frac{\tau_{\varepsilon l}^{P}}{\tau_{\sigma l}^{P}} \right) \right] - 2\mu \left[1 - \sum_{l=1}^{L} \left(1 - \frac{\tau_{\varepsilon l}^{S}}{\tau_{\sigma l}^{S}} \right) \right] \right\} \\
& \times \left(\frac{\partial v_x}{\partial x} + \frac{\partial v_z}{\partial z} \right) + 2\mu \left[1 - \sum_{l=1}^{L} \left(1 - \frac{\tau_{\varepsilon l}^{S}}{\tau_{\sigma l}^{S}} \right) \right] \frac{\partial v_z}{\partial z} + \\
& \sum_{l=1}^{L} \pi \frac{1}{\tau_{\sigma l}^{P}} \left(1 - \frac{\tau_{el}^{P}}{\tau_{\sigma l}^{P}} \right) e^{-t/\tau_{\sigma l}^{P}} \theta(t) * \left(\frac{\partial v_x}{\partial x} + \frac{\partial v_z}{\partial z} \right) + \\
& \sum_{l=1}^{L} -2\mu \frac{1}{\tau_{\sigma l}^{S}} \left(1 - \frac{\tau_{\varepsilon l}^{S}}{\tau_{\sigma l}^{S}} \right) e^{-t/r_{\sigma l}^{S}} \theta(t) * \frac{\partial v_x}{\partial x},
\end{aligned} \quad (21)$$

$$\begin{aligned}
\dot{\sigma}_{xz} =& \dot{M} * (\partial_z v_x + \partial_x v_z) \\
=& \mu \left[1 - \sum_{l=1}^{L} \left(1 - \frac{\tau_{\varepsilon l}^{S}}{\tau_{\sigma l}^{S}} \right) \right] \left(\frac{\partial v_z}{\partial x} + \frac{\partial v_x}{\partial z} \right) + \sum_{l=1}^{L} \mu \frac{1}{\tau_{\sigma l}^{S}} \left(1 - \frac{\tau_{\varepsilon l}^{S}}{\tau_{\sigma l}^{S}} \right) e^{-t/\tau_{\sigma l}^{S}} \theta(t) * \left(\frac{\partial v_z}{\partial x} + \frac{\partial v_x}{\partial z} \right),
\end{aligned} \quad (22)$$

The convolution factor in each formula can be considered as the memory variables, which could be subtracted as $\left[\tau_{xxl}^{P}, \tau_{xxl}^{S}, \tau_{zzl}^{P}, \tau_{zzl}^{S}, \tau_{xzl}^{S}\right]$:

$$\tau_{xxl}^{P} = \pi \frac{1}{\tau_{\sigma l}^{P}} \left(1 - \frac{\tau_{\varepsilon l}^{P}}{\tau_{\sigma l}^{P}} \right) e^{-t/\tau_{\sigma l}^{P}} \theta(t) * \left(\frac{\partial v_x}{\partial x} + \frac{\partial v_z}{\partial z} \right), \quad (23)$$

$$\tau_{xxl}^{S} = -2\mu \frac{1}{\tau_{\sigma l}^{S}} \left(1 - \frac{\tau_{\varepsilon l}^{S}}{\tau_{\sigma l}^{S}}\right) e^{-t/\tau_{\sigma l}^{S}} \theta(t) * \frac{\partial v_z}{\partial z}, \tag{24}$$

$$\tau_{zzl}^{P} = \pi \frac{1}{\tau_{\sigma l}^{p}} \left(1 - \frac{\tau_{\varepsilon l}^{P}}{\tau_{\sigma l}^{p}}\right) e^{-t/\tau_{\sigma l}^{P}\theta} (t) * \left(\frac{\partial v_x}{\partial x} + \frac{\partial v_z}{\partial z}\right), \tag{25}$$

$$\tau_{zzl}^{S} = -2\mu \frac{1}{\tau_{\sigma l}^{S}} \left(1 - \frac{\tau_{\varepsilon l}^{S}}{\tau_{\sigma l}^{S}}\right) e^{-t/\tau_{\sigma l}^{S}} \theta(t) * \frac{\partial v_x}{\partial x}, \tag{26}$$

$$\tau_{xzl}^{S} = \mu \frac{1}{\tau_{\sigma l}^{S}} \left(1 - \frac{\tau_{\varepsilon l}^{S}}{\tau_{\sigma l}^{S}}\right) e^{-t/\tau_{\sigma l}^{S}} \theta(t) * \left(\frac{\partial v_z}{\partial x} + \frac{\partial v_x}{\partial z}\right), \tag{27}$$

Form formula (15) and (16) we can see that τ_{xxl}^{P} and τ_{zzl}^{P} has the same mathematical expression. Thus we fuse them together as τ_{l}^{P}:

$$\tau_{l}^{P} = \tau_{xxl}^{P} = \tau_{zzl}^{P} = \pi \frac{1}{\tau_{\sigma l}^{P}} \left(1 - \frac{\tau_{\varepsilon l}^{P}}{\tau_{\sigma l}^{P}}\right) e^{-t/\tau_{\sigma l}^{P}} \theta(t) * \left(\frac{\partial v_x}{\partial x} + \frac{\partial v_z}{\partial z}\right), \tag{28}$$

Take time partial derivative of equation (15) we have :

$$\begin{aligned} \dot{\tau_{l}^{P}} &= \left(\frac{\partial v_x}{\partial x} + \frac{\partial v_z}{\partial z}\right) * \left[\pi \frac{-1}{\tau_{\sigma l}^{P2}} \left(1 - \frac{\tau_{\varepsilon l}^{P}}{\tau_{\sigma l}^{P}}\right) e^{-t/\tau_{\sigma l}^{P}} \theta(t) + \pi \frac{1}{\tau_{\sigma l}^{P}} \left(1 - \frac{\tau_{\varepsilon l}^{P}}{\tau_{\sigma l}^{P}}\right) e^{-t/\tau_{\sigma l}^{P}} \delta(t)\right] \\ &= -\frac{1}{\tau_{\sigma l}^{p}} \tau_{l}^{P} - \frac{1}{\tau_{\sigma l}^{p}} \pi \left(\frac{\tau_{\varepsilon l}^{P}}{\tau_{\sigma l}^{P}} - 1\right) \left(\frac{\partial v_x}{\partial x} + \frac{\partial v_z}{\partial z}\right), \end{aligned} \tag{29}$$

Similarly, we have:

$$\dot{\tau}_{xxl}^{S} = -\frac{1}{\tau_{\sigma l}^{S}} \left[r_{xxl}^{S} - 2\mu \left(\frac{\tau_{\varepsilon l}^{S}}{\tau_{\sigma l}^{S}} - 1\right) \frac{\partial v_x}{\partial x}\right], \tag{30}$$

$$\dot{\tau}_{zzl}^{S} = -\frac{1}{\tau_{\sigma l}^{S}} \left[r_{zzl}^{S} - 2\mu \left(\frac{\tau_{\varepsilon l}^{S}}{\tau_{\sigma l}^{S}} - 1\right) \frac{\partial v_z}{\partial z}\right], \tag{31}$$

$$\dot{\tau}_{xzl}^{S} = -\frac{1}{\tau_{\sigma l}^{S}} \left[r_{xzl}^{S} + \mu \left(\frac{\tau_{\varepsilon l}^{S}}{\tau_{\sigma l}^{S}} - 1\right) \left(\frac{\partial v_z}{\partial x} + \frac{\partial v_x}{\partial z}\right)\right], \tag{32}$$

Following the constitutive relationship and the momentum conservation equation. The 2D viscoelastic wave equations are:

$$\rho \dot{v}_x = \frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{xz}}{\partial x} + f_x, \tag{33}$$

$$\rho \dot{v}_z = \frac{\partial \sigma_{xz}}{\partial z} + \frac{\partial \sigma_{zz}}{\partial z} + f_z, \tag{34}$$

$$\dot{\sigma}_{xx} = \pi \left[1 - \sum_{l=1}^{L} \left(1 - \frac{\tau_{\varepsilon l}^{P}}{\tau_{\sigma l}^{P}}\right)\right] \left(\frac{\partial v_x}{\partial x} + \frac{\partial v_z}{\partial z}\right) - 2\mu \left[1 - \sum_{l=1}^{L} \left(1 - \frac{\tau_{\varepsilon l}^{S}}{\tau_{\sigma l}^{S}}\right)\right] \frac{\partial v_z}{\partial z} + R^{P} + R_{xx}^{S}, \tag{35}$$

$$\dot{\sigma}_{zz} = \pi \left[1 - \sum_{l=1}^{L}\left(1 - \frac{\tau_{\varepsilon l}^{P}}{\tau_{\sigma l}^{P}}\right)\right]\left(\frac{\partial v_x}{\partial x} + \frac{\partial v_z}{\partial z}\right) - 2\mu\left[1 - \sum_{l=1}^{L}\left(1 - \frac{\tau_{\varepsilon l}^{S}}{\tau_{\sigma l}^{S}}\right)\right]\frac{\partial v_x}{\partial x} + R^P + R_{zz}^S, \tag{36}$$

$$\dot{\sigma}_{xz} = \mu\left[1 - \sum_{l=1}^{L}\left(1 - \frac{\tau_{\varepsilon l}^{S}}{\tau_{\sigma l}^{Sl}}\right)\right]\left(\frac{\partial v_z}{\partial x} + \frac{\partial v_x}{\partial z}\right) + R_{xz}^S, \tag{37}$$

$$R^P = \sum_{l=1}^{L}\tau_l^P, \tag{38}$$

$$R_{xx}^S = \sum_{l=1}^{L}\tau_{xxl}^S, \tag{39}$$

$$R_{zz}^S = \sum_{l=1}^{L}\tau_{zzl}^S, \tag{40}$$

$$R_{xz}^S = \sum_{l=1}^{L}\tau_{xzl}^S, \tag{41}$$

Thus, when $L = 1$ the viscoelastic wave equation is:

$$\begin{cases} \dfrac{\partial v_x}{\partial t} = \dfrac{1}{\rho}\left(\dfrac{\partial \sigma_{xx}}{\partial x} + \dfrac{\partial \sigma_{xy}}{\partial y}\right) \\ \dfrac{\partial v_y}{\partial t} = \dfrac{1}{\rho}\left(\dfrac{\partial \sigma_{xy}}{\partial x} + \dfrac{\partial \sigma_{yy}}{\partial y}\right) \\ \dfrac{\partial \sigma_{xy}}{\partial t} = \mu\dfrac{\tau_\varepsilon^s}{\tau_\sigma}\left(\dfrac{\partial v_x}{\partial y} + \dfrac{\partial v_y}{\partial x}\right) + r_{xy} \\ \dfrac{\partial \sigma_{xx}}{\partial t} = \pi\dfrac{\tau_\varepsilon^p}{\tau_\sigma}\left(\dfrac{\partial v_x}{\partial x} + \dfrac{\partial v_y}{\partial y}\right) - 2\mu\dfrac{\tau_\varepsilon^s}{\tau_\sigma}\dfrac{\partial v_y}{\partial y} + r_{xx} \\ \dfrac{\partial \sigma_{yy}}{\partial t} = \pi\dfrac{\tau_\varepsilon^p}{\tau_\sigma}\left(\dfrac{\partial v_x}{\partial x} + \dfrac{\partial v_y}{\partial y}\right) - 2\mu\dfrac{\tau_\varepsilon^s}{\tau_\sigma}\dfrac{\partial v_x}{\partial x} + r_{yy} \\ \dfrac{\partial r_{xy}}{\partial t} = -\dfrac{1}{\tau_\sigma}\left(r_{xy} + \mu\left(\dfrac{\tau_\varepsilon^s}{\tau_\sigma} - 1\right)\left(\dfrac{\partial v_x}{\partial y} + \dfrac{\partial v_y}{\partial x}\right)\right) \\ \dfrac{\partial r_{xx}}{\partial t} = -\dfrac{1}{\tau_\sigma}\left(r_{xx} + \pi\left(\dfrac{\tau_\varepsilon^p}{\tau_\sigma} - 1\right)\left(\dfrac{\partial v_x}{\partial x} + \dfrac{\partial v_y}{\partial y}\right) - 2\mu\left(\dfrac{\tau_\varepsilon^s}{\tau_\sigma} - 1\right)\dfrac{\partial v_y}{\partial y}\right) \\ \dfrac{\partial r_{yy}}{\partial t} = -\dfrac{1}{\tau_\sigma}\left(r_{yy} + \pi\left(\dfrac{\tau_\varepsilon^p}{\tau_\sigma} - 1\right)\left(\dfrac{\partial v_x}{\partial x} + \dfrac{\partial v_y}{\partial y}\right) - 2\mu\left(\dfrac{\tau_\varepsilon^s}{\tau_\sigma} - 1\right)\dfrac{\partial v_x}{\partial x}\right) \end{cases}, \tag{42}$$

REFERENCES

Belahi, T., Fuji, N., and Singh, S., 2015, Elastic versus viscoelastic full waveform inversion of near-offset and wide-angle data in the presence of attenuation, *in* 77th EAGE Conference and Exhibition 2015.

Belahi, T., Singh, S., and Fuji, N., 2016, Viscoelastic full waveform inversion of sea bottom long offset seismic data in presence of attenuation, *in* 78th EAGE Conference and Exhibition 2016.

Bohlen, T., 2002, Parallel 3-d viscoelastic finite difference seismic modelling: Computers & Geosciences, **28**, No. 8, 887–899.

Brossier, R., Operto, S., and Virieux, J., 2010, Which data residual norm for robust elastic frequency-domain full waveform inversion?: Geophysics, **75**, No. 3, R37–R46.

Day, S., and Minster, J., 1984, Numerical simulation of wavefields using a pade approximation method: Geophys. JR astr. Soc, 78–105.

Djikpéssé, H. A., and Tarantola, A., 1999, Multiparameter l1 norm waveform fitting: Interpretation of gulf of mexico reflection seismograms: Geophysics, **64**, No. 4, 1023–1035.

Emmerich, H., and Korn, M., 1987, Incorporation of attenuation into time-domain computations of seismic wave fields: Geophysics, **52**, No. 9, 1252–1264.

Fabien-Ouellet, G., Gloaguen, E., and Giroux, B., 2016, The adjoint state method for the viscoelastic wave equation in the velocity-stress formulation, *in* 78th EAGE Conference and Exhibition 2016.

Fabien-Ouellet, G., Gloaguen, E., and Giroux, B., 2017, Time-domain seismic modeling in viscoelastic media for full waveform inversion on heterogeneous computing platforms with opencl: Computers & Geosciences, **100**, 142–155.

Groos, L., Schäfer, M., Forbriger, T., and Bohlen, T., 2012, On the significance of viscoelasticity in a 2d full waveform inversion of shallow seismic surface waves, *in* 74th EAGE Conference and Exhibition incorporating EUROPEC 2012.

Lin, Y., and Wu, Y., 2018, Inversionnet: A real-time and accurate full waveform inversion with convolutional neural network: The Journal of the Acoustical Society of America, **144**, No. 3, 1683–1683.

Liu, H.-P., Anderson, D. L., and Kanamori, H., 1976, Velocity dispersion due to anelasticity; implications for seismology and mantle composition: Geophysical Journal International, **47**, No. 1, 41–58.

Mosser, L., Dubrule, O., and Blunt, M., 2018, Stochastic seismic waveform inversion using generative adversarial networks as a geological prior, *in* First EAGE/PESGB Workshop Machine Learning.

Pyun, S., Son, W., and Shin, C., 2009, Frequency-domain waveform inversion using an l 1-norm objective function: Exploration Geophysics, **40**, No. 2, 227–232.

Robertsson, J. O., Blanch, J. O., and Symes, W. W., 1994, Viscoelastic finite-difference modeling: Geophysics, **59**, No. 9, 1444–1456.

Sun, H., and Demanet, L., 2018, Low frequency extrapolation with deep learning, *in* SEG Technical Program Expanded Abstracts 2018, Society of Exploration Geophysicists, 2011–2015.

Trinh, P.-T., Brossier, R., Métivier, L., and Virieux, J., 2018, Data-oriented strategy and v p/v s model-constraint for simultaneous v p and v s reconstruction in 3d visco-elastic fwi: Application to the seam ii foothill dataset, *in* SEG Technical Program Expanded Abstracts 2018, Society of Exploration Geophysicists, 1213–1217.

Yang, F., and Ma, J., 2019, Deep-learning inversion: a next generation seismic velocity-model building method: Geophysics, **84**, No. 4, 1–133.

Yang, P., Brossier, R., Métivier, L., and Virieux, J., 2016, A review on the systematic formulation of 3-d multiparameter full waveform inversion in viscoelastic medium: Geophysical Journal International, **207**, No. 1, 129–149.