ArjunAir: Updating and parallelizing an existing time domain electromagnetic inversion program

Patrick Belliveau*, Dr. Colin Farquharson, Dr. Ronald Haynes
Memorial University of Newfoundland

The Forward Model

Forward problem overview

ArjunAir is a 2.5D airborne electromagnetic inversion program developed by CSIRO in partnership with the AMIRA consortium [4]. It inverts for a 2D conductivity distribution. It solves for the along-strike components of the secondary electric and magnetic fields using a nodal finite element method with isoparametric quadrilateral elements. The secondary field Maxwell equations in the frequency domain are

\[ \nabla \times \left( \nabla \times \mathbf{E} \right) = -4\pi \mathbf{J} \]

where, \( \mathbf{J} \) is the current density. These integrals were originally computed in ArjunAir but inversion results are still too dependent on initial guess. The use of minimum structure inversion should improve inversion results without sacrificing speed. A minimum structure inversion code using the ArjunAir forward solver has been mostly written.

Improvements

Parallel Computing Paradigms

Inversion algorithm

ArjunAir’s inversion algorithm is a variant of the Levenberg-Marquardt algorithm. It uses singular value damping to stabilize the inversion. The singular value decomposition is expensive both to compute and to store in memory. We formulated the inverse problem using the standard Levenberg-Marquardt algorithm. It uses singular value damping to stabilize the inversion. The singular value decomposition is expensive both to compute and to store in memory. We formulated the inverse problem using the standard Levenberg-Marquardt algorithm.

We have significantly improved ArjunAir run times through the use of parallel computing and more efficient sequential algorithms. It makes ArjunAir a more useful tool, even for code but inversion results are still too dependent on initial guess. The use of minimum structure inversion should improve inversion results without sacrificing speed. A minimum structure inversion code using the ArjunAir forward solver has been mostly written.

Inversion algorithm

ArjunAir’s inversion algorithm is a variant of the Levenberg-Marquardt algorithm. It uses singular value damping to stabilize the inversion. The singular value decomposition is expensive both to compute and to store in memory. We formulated the inverse problem using the standard Levenberg-Marquardt algorithm.

\[ \mathbf{J} = \mathbf{M}^T \mathbf{P} \]

where \( \mathbf{M} \) is the primary field component and \( \mathbf{P} \) is the primary field. We need it in the spatial wavenumber domain. We compute Fourier transform numerically. E.g.,

\[ \mathbf{F} = \frac{1}{N} \sum_{k=0}^{N-1} \mathbf{E}(k) e^{-2\pi i k x / N} \]

where, \( x \) is the spatial coordinate. These integrals were originally computed in ArjunAir by digital filtering at every mesh point for every transmitter. However, depend only on the primary field and not on the across strike coordinates individually. Compute integral at set of wavenumbers and interpolate.

Parallel Computing Paradigms

We used MPI for distributed memory parallelization and OpenMP for shared memory. All code was written in Fortran.

The frequency domain primary field is

\[ \mathbf{E} = \mathbf{M}^T \mathbf{P} \]

Computing the primary field

The frequency domain primary electric field is

\[ \mathbf{E} = \mathbf{M}^T \mathbf{P} \]

We need it in the spatial wavenumber domain. We compute Fourier transform numerically. E.g.,

\[ \mathbf{F} = \frac{1}{N} \sum_{k=0}^{N-1} \mathbf{E}(k) e^{-2\pi i k x / N} \]

where, \( x \) is the spatial coordinate. These integrals were originally computed in ArjunAir by digital filtering at every mesh point for every transmitter. However, depend only on the primary field and not on the across strike coordinates individually. Compute integral at set of wavenumbers and interpolate.

Improvements

Parallel Computing Paradigms

We used MPI for distributed memory parallelization and OpenMP for shared memory. All code was written in Fortran.

Improvements

Parallel Computing Paradigms

We used MPI for distributed memory parallelization and OpenMP for shared memory. All code was written in Fortran.

The frequency domain primary field is

\[ \mathbf{E} = \mathbf{M}^T \mathbf{P} \]

Computing the primary field

The frequency domain primary electric field is

\[ \mathbf{E} = \mathbf{M}^T \mathbf{P} \]

We need it in the spatial wavenumber domain. We compute Fourier transform numerically. E.g.,

\[ \mathbf{F} = \frac{1}{N} \sum_{k=0}^{N-1} \mathbf{E}(k) e^{-2\pi i k x / N} \]

where, \( x \) is the spatial coordinate. These integrals were originally computed in ArjunAir by digital filtering at every mesh point for every transmitter. However, depend only on the primary field and not on the across strike coordinates individually. Compute integral at set of wavenumbers and interpolate.

Parallel Computing Paradigms

We used MPI for distributed memory parallelization and OpenMP for shared memory. All code was written in Fortran.

Improvements

Parallel Computing Paradigms

We used MPI for distributed memory parallelization and OpenMP for shared memory. All code was written in Fortran.

The frequency domain primary field is

\[ \mathbf{E} = \mathbf{M}^T \mathbf{P} \]

Computing the primary field

The frequency domain primary electric field is

\[ \mathbf{E} = \mathbf{M}^T \mathbf{P} \]

We need it in the spatial wavenumber domain. We compute Fourier transform numerically. E.g.,

\[ \mathbf{F} = \frac{1}{N} \sum_{k=0}^{N-1} \mathbf{E}(k) e^{-2\pi i k x / N} \]

where, \( x \) is the spatial coordinate. These integrals were originally computed in ArjunAir by digital filtering at every mesh point for every transmitter. However, depend only on the primary field and not on the across strike coordinates individually. Compute integral at set of wavenumbers and interpolate.

Parallel Computing Paradigms

We used MPI for distributed memory parallelization and OpenMP for shared memory. All code was written in Fortran.

Improvements

Parallel Computing Paradigms

We used MPI for distributed memory parallelization and OpenMP for shared memory. All code was written in Fortran.

The frequency domain primary field is

\[ \mathbf{E} = \mathbf{M}^T \mathbf{P} \]

Computing the primary field

The frequency domain primary electric field is

\[ \mathbf{E} = \mathbf{M}^T \mathbf{P} \]

We need it in the spatial wavenumber domain. We compute Fourier transform numerically. E.g.,

\[ \mathbf{F} = \frac{1}{N} \sum_{k=0}^{N-1} \mathbf{E}(k) e^{-2\pi i k x / N} \]

where, \( x \) is the spatial coordinate. These integrals were originally computed in ArjunAir by digital filtering at every mesh point for every transmitter. However, depend only on the primary field and not on the across strike coordinates individually. Compute integral at set of wavenumbers and interpolate.

Parallel Computing Paradigms

We used MPI for distributed memory parallelization and OpenMP for shared memory. All code was written in Fortran.

Improvements

Parallel Computing Paradigms

We used MPI for distributed memory parallelization and OpenMP for shared memory. All code was written in Fortran.

The frequency domain primary field is

\[ \mathbf{E} = \mathbf{M}^T \mathbf{P} \]

Computing the primary field

The frequency domain primary electric field is

\[ \mathbf{E} = \mathbf{M}^T \mathbf{P} \]

We need it in the spatial wavenumber domain. We compute Fourier transform numerically. E.g.,

\[ \mathbf{F} = \frac{1}{N} \sum_{k=0}^{N-1} \mathbf{E}(k) e^{-2\pi i k x / N} \]

where, \( x \) is the spatial coordinate. These integrals were originally computed in ArjunAir by digital filtering at every mesh point for every transmitter. However, depend only on the primary field and not on the across strike coordinates individually. Compute integral at set of wavenumbers and interpolate.

Parallel Computing Paradigms

We used MPI for distributed memory parallelization and OpenMP for shared memory. All code was written in Fortran.

Improvements

Parallel Computing Paradigms

We used MPI for distributed memory parallelization and OpenMP for shared memory. All code was written in Fortran.

The frequency domain primary field is

\[ \mathbf{E} = \mathbf{M}^T \mathbf{P} \]

Computing the primary field

The frequency domain primary electric field is

\[ \mathbf{E} = \mathbf{M}^T \mathbf{P} \]

We need it in the spatial wavenumber domain. We compute Fourier transform numerically. E.g.,

\[ \mathbf{F} = \frac{1}{N} \sum_{k=0}^{N-1} \mathbf{E}(k) e^{-2\pi i k x / N} \]

where, \( x \) is the spatial coordinate. These integrals were originally computed in ArjunAir by digital filtering at every mesh point for every transmitter. However, depend only on the primary field and not on the across strike coordinates individually. Compute integral at set of wavenumbers and interpolate.

Parallel Computing Paradigms

We used MPI for distributed memory parallelization and OpenMP for shared memory. All code was written in Fortran.

Improvements

Parallel Computing Paradigms

We used MPI for distributed memory parallelization and OpenMP for shared memory. All code was written in Fortran.

The frequency domain primary field is

\[ \mathbf{E} = \mathbf{M}^T \mathbf{P} \]

Computing the primary field

The frequency domain primary electric field is

\[ \mathbf{E} = \mathbf{M}^T \mathbf{P} \]

We need it in the spatial wavenumber domain. We compute Fourier transform numerically. E.g.,

\[ \mathbf{F} = \frac{1}{N} \sum_{k=0}^{N-1} \mathbf{E}(k) e^{-2\pi i k x / N} \]

where, \( x \) is the spatial coordinate. These integrals were originally computed in ArjunAir by digital filtering at every mesh point for every transmitter. However, depend only on the primary field and not on the across strike coordinates individually. Compute integral at set of wavenumbers and interpolate.

Parallel Computing Paradigms

We used MPI for distributed memory parallelization and OpenMP for shared memory. All code was written in Fortran.