## 2维时间域全波形反演程序说明

程序流程图如下所示：

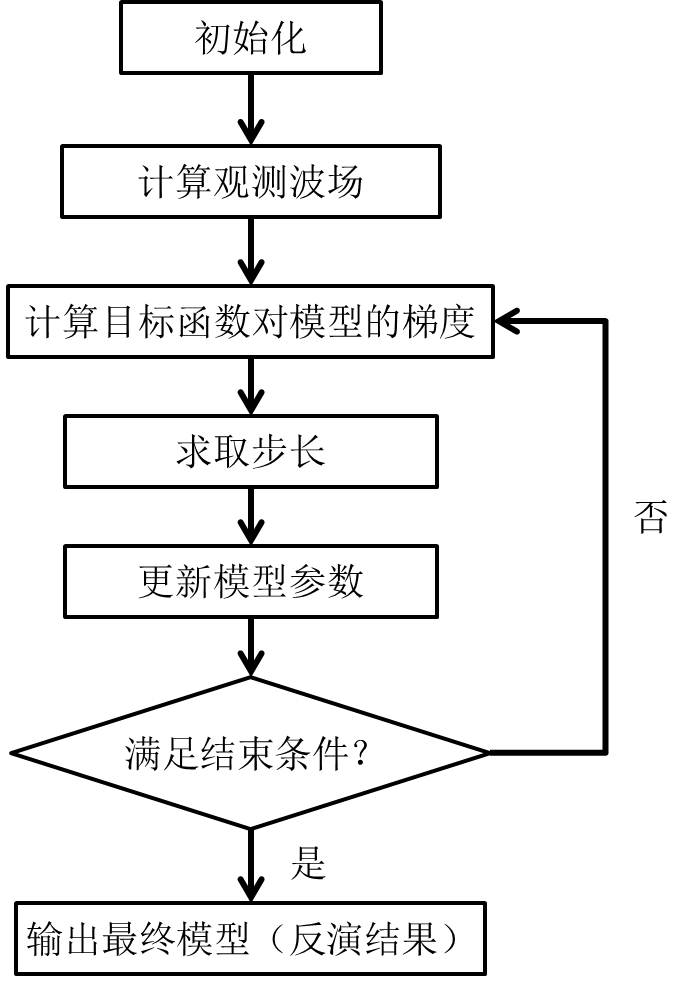


图1：程序流程图

1. 初始化

初始化部分，我们为整个程序中所使用的变量申请了空间，对反演相关参数进行了赋值。具体内容如下：

|  |  |
| --- | --- |
| Pa->dt | 采样间隔 |
| Pa->dx | 有限差分网格横向网格间隔 |
| Pa->dz | 有限差分网格纵向网格间隔 |
| Pa->f0 | Ricker子波的主频 |
| Pa->Nt | 正演及反演过程中采样点数 |
| Pa->Nx | 有限差分横向网格数 |
| Pa->Nz | 有限差分纵向网格数 |
| Pa->PMLx | 横向NPML边界网格数 |
| Pa->PMLz | 纵向NPML边界网格数 |
| ip->ShotN | 整个反演过程中使用的炮数 |
| ip->Iter | 反演迭代的次数 |
| ip->Alpha | 反演迭代步长（初始化为0） |
| ip->St[is].rn | 第is炮对应的检波器个数 |
| ip->St[is].s.Sx | 第is炮对应的震源的横坐标 |
| ip->St[is].s.Sz | 第is炮对应的震源的纵坐标 |
| ip->St[is].re[m].Rx | 第is炮中第m个检波器的横坐标 |
| ip->St[is].re[m].Rz | 第is炮中第m个检波器的纵坐标 |
| ip->TrueVp | 真实模型（速度的平方） |
| ip-> CurrVp | 初始模型（速度的平方） |

除了上述变量赋值，我们还调用了如下三个函数：

|  |  |
| --- | --- |
| MallocVariables(\*Pa, ip, plan); | 为全局变量开辟空间 |
| GenerateNPML(\*Pa, plan); | 为NPML（非分裂完全匹配曾）参数赋值 |
| SetCoord(Pa, plan->h\_Coord); | 为有效边界存储策略中需要存储的波场坐标赋值 |

1. 计算观测波场

程序中，函数CalTrueWF(\*Pa, ip, plan, sgs\_t)；完成的是计算观测波场的功能。具体来说，函数采用离散差分来代替微分，求取不同时刻点上波场的值，然后记录下在检波器位置上的波场值，形成炮集，存储在sgs\_t中。此函数中调用了如下的一些子函数：

|  |  |
| --- | --- |
| StepShotGather | 记录it时刻第is炮所对用的检波器位置上的波场值 |
| StepPHIU | 在it时刻更新波场U的卷积项 |
| StepVW | 在it时刻更新V和W两个波场 |
| StepPHIVW | 在it时刻更新波场V和W的卷积项 |
| StepU | 在it时刻更新波场U |
| AddSource | 波场加震源 |

1. 计算目标函数对模型的梯度

程序中CalGrad函数完成的是计算目标函数对模型的梯度的功能。其中，首先计算在当前模型下，对应的计算波场。然后按照理论，进行正传波场的反传（逆着时间传）以及残差波场的正传，并伴随着波的传播计算出梯度。此函数中调用的子函数如下所示：

|  |  |
| --- | --- |
| StepShotGather | 记录it时刻第is炮所对用的检波器位置上的波场值 |
| StepRecordU | 记录在it时刻有效边界内的波场值 |
| StepPHIU | 在it时刻更新波场U的卷积项 |
| StepVW | 在it时刻更新V和W两个波场 |
| StepPHIVW | 在it时刻更新波场V和W的卷积项 |
| StepU | 在it时刻更新波场U |
| AddSource | 波场加震源 |
| StepResidual | 计算观测波场和计算波场的残差 |
| StepCal2Der | 求取it时刻正传波场对时间的2阶导 |
| StepReplaceU | 用记录的有效边界内的波场取代当前波场的值 |
| StepRtVW | 一步逆时间更新V和W |
| StepRtU | 一步逆时间更新U |
| StepCalGrad | 一步求取梯度 |

1. 求取步长

程序中CalStepLength函数实现的是求取步长的功能。其本质是产生一个试探模型参数，使用此模型传播波场，得到相对应的波场。利用此波场和当前波场的残差，按照公式计算出步长。

1. 更新模型参数

在得到目标函数对模型的梯度以及步长后，程序中PreProcess函数实现了对当前模型参数进行更新，得到新的模型参数。

1. 终止条件

在程序中，我们采取给定固定迭代次数为终止条件。当算法迭代到固定的次数后，程序终止。

1. 输出最终模型

在程序中，我们将最终反演结果通过WriteData函数写成“InvertedVp.sgy”文件。