```
Algorithm solve_fde(navdata, method="residual", remove_outliers=False,
              max_faults=None, threshold=None, verbose=False,
              **kwargs)
    if method == "residual"  navdata = fde_greedy_residual
    elif method == "edm"      navdata = fde_edm

Algorithm fde_greedy_residual
for navdata_subset in time //对于每个时间点的数据子集
    receiver_state = solve_wls(navdata_subset) //先计算一个wls
    solve_residuals(navdata_subset, receiver_state, inplace=True)
    chi_square = _residual_chi_square(navdata_subset, receiver_state)
    while chi_square > threshold
        _residual_exclude(navdata_subset,receiver_state)//计算归一化残差值越大的卫星测
量值
         fault_idx = np.argsort(normalized_residual)[-1]//记录该卫星编号
         navdata_subset.remove(cols=[fault_idx], inplace=True)//删除该卫星测量数据
         receiver_state = solve_wls(navdata_subset)//重新计算wls
         solve_residuals(navdata_subset, receiver_state, inplace=True)//计算残差数
据
         chi_square = _residual_chi_square(navdata_subset, receiver_state)//利用残
差数据计算卡方统计量
return navdata


Algorithm fde_edm
for navdata_subset in time  # 对每个时间点的数据子集
    # 构建欧氏距离矩阵
    edm = _edm_from_satellites_ranges(sv_m, corr_pr_m)
    detection_statistic = np.inf

    while detection_statistic > threshold
        # 计算检测统计量和奇异值分解
        edm_detect = np.delete(np.delete(edm, fault_idxs, 0), fault_idxs, 1)
        detection_statistic_detect, svd_u, _, _ =
_edm_detection_statistic(edm_detect)
        detection_statistic = detection_statistic_detect[0]

        if detection_statistic < threshold
            break

        # 通过分析奇异向量识别可疑故障卫星
        u3_suspects = np.argsort(np.abs(svd_u[:,3]))[::-1][:nci]
        u4_suspects = np.argsort(np.abs(svd_u[:,4]))[::-1][:nci]
        suspects = u3_suspects + u4_suspects

        # 计算每个可疑卫星的出现次数
        counts = {i:suspects.count(i) for i in suspects if i != 0}
        fault_suspects = [[i] for i,v in counts.items() if v == 2]

        # 对每个可疑故障卫星计算移除后的检测统计量
        stacked_edms = [np.delete(np.delete(edm, fault_idxs+i, 0),
                                  fault_idxs+i, 1) for i in fault_suspects]
        detection_statistic_exclude, _, _, _ =
_edm_detection_statistic(stacked_edms)
```

```
        # 选择使检测统计量最小的移除方案
        min_idx = np.argmin(detection_statistic_exclude)
        fault_idxs += fault_suspects[min_idx]
        detection_statistic = detection_statistic_exclude[min_idx]

        # 检查是否达到最大故障数限制
        if len(fault_idxs) >= max_faults
            break

return navdata
```