

Implementation of poloidal density variation in collision operator

Aylwin Iantchenko

October 5, 2017

The aim of this document is to explain the modifications in **SFINCS** that have been done, in order to include poloidal density variation in the collision operator (in both full Fokker-Planck and pure pitch angle scattering operator options).

As is explained in Ref. [1] poloidal density variation can be included by modifying the lowest order distribution function

$$f_{Ms}(\psi) \rightarrow f_{0s} = f_{Ms}(\psi) e^{-Z_s e \Phi_1(\theta, \zeta)/T_s},$$

where $f_M(\psi)$ is the flux-function Maxwellian, Φ_1 are first order variation of the electrostatic potential, and Z_s, T_s are the charge and temperature of species s respectively. To implement this change one may simply modify the species density accordingly

$$n_s \rightarrow n_s e^{-Ze\Phi_1(\theta, \zeta)/T}, \quad (1)$$

in the original equations for the various terms in the collision operator, which are described in Ref. [2]. The linearised collision operator takes the new form

$$\begin{aligned} C_{ab}^{L:f0} = & C_{ab} \{f_{aM}, f_{bM}\} e^{-(Z_a/T_a + Z_b/T_b) e \Phi_1(\theta, \zeta)} + C_{ab} \{f_{a1}, f_{bM}\} e^{-Z_b/T_b e \Phi_1(\theta, \zeta)} + \\ & + C_{ab} \{f_{aM}, f_{b1}\} e^{-Z_a/T_a e \Phi_1(\theta, \zeta)}, \end{aligned} \quad (2)$$

which describes collisions between species a and b . In normalised **SFINCS** units this becomes

$$\begin{aligned} \hat{C}_{ab}^{L:f0} = & \hat{C}_{ab} \{f_{aM}, f_{bM}\} e^{-(\hat{Z}_a/\hat{T}_a + \hat{Z}_b/\hat{T}_b) \alpha \hat{\Phi}_1(\theta, \zeta)} + \hat{C}_{ab} \{f_{a1}, f_{bM}\} e^{-\hat{Z}_b/\hat{T}_b \alpha \hat{\Phi}_1(\theta, \zeta)} + \\ & + \hat{C}_{ab} \{f_{aM}, f_{b1}\} e^{-\hat{Z}_a/\hat{T}_a \alpha \hat{\Phi}_1(\theta, \zeta)}. \end{aligned} \quad (3)$$

The first term on the RHS of Eq. (3) is the *temperature equilibration term*. Since this term does not include f_{1s} it has to be treated differently compared to the other two cases, as will be seen shortly. In the code, the calculation of the residual has to be modified to include the factors specified in Eq. (3).

The contribution to the Jacobian are $\partial \hat{C}_{ab}^{L:f0} / \partial \hat{f}_{1s}$ and $\partial \hat{C}_{ab}^{L:f0} / \partial \hat{\Phi}_1$. The first is the same as in Eq. (3) except that the collision operator does not act on the distribution function. In the second case we get

$$\begin{aligned} \frac{\partial \hat{C}_{ab}^{L:f0}}{\partial \hat{\Phi}_1} = & - \left(\hat{Z}_a / \hat{T}_a + \hat{Z}_b / \hat{T}_b \right) \alpha \hat{C}_{ab} \{f_{aM}, f_{bM}\} e^{-(\hat{Z}_a / \hat{T}_a + \hat{Z}_b / \hat{T}_b) \alpha \hat{\Phi}_1(\theta, \zeta)} \\ & - \hat{Z}_b / \hat{T}_b \alpha \hat{C}_{ab} \{f_{a1}, f_{bM}\} e^{-\hat{Z}_b / \hat{T}_b \alpha \hat{\Phi}_1(\theta, \zeta)} - \hat{Z}_a / \hat{T}_a \alpha \hat{\Phi}_1(\theta, \zeta) \hat{C}_{ab} \{f_{aM}, f_{b1}\} e^{-\hat{Z}_a / \hat{T}_a \alpha \hat{\Phi}_1(\theta, \zeta)}. \end{aligned} \quad (4)$$

Apart from the different pre-factor, the term in Eq. (4) differs from term in $\partial \hat{C}_{ab}^{L:f0} / \partial \hat{f}_{1s}$ by acting on the first order distribution function f_{1s} . For example, in the first case, terms such as $\partial f_{1s} / \partial x_s$ become

$$\frac{1}{\partial f_{1s}} \frac{\partial f_{1s}}{\partial x_s} = \frac{\partial}{\partial x_s},$$

whereas in the second case, when we take the derivative with respect to Φ_1 , we have to include the derivative $\partial f_{1s} / \partial x_s$ (and $\partial f_{0s} / \partial x_s$ for temperature equilibration), as well.

Implementation in the Code

To implement Eq. (3) and Eq. (4) we need to modify the evaluation of the residual and the Jacobian. The residual is evaluated in `evaluateResidual.F90` and the Jacobian in `evaluateJacobian.F90`. Both these two routines call the functions in `PopulateMatrix.F90`, where the actual assembly of the residual and Jacobian matrices are done. As a consequence of this structure, we only need to modify `PopulateMatrix.F90` in order to include poloidal density variation in the collision operator.

Residual

In the residual we have to include the extra pre-factor appearing in Eq. (3). Since this factor is species specific, we have to keep track of which species density is appearing in the equations. The pitch-angle scattering and the $\hat{C}_{ab}^E \{f_{a1}\}$ parts of the collision operator (see Ref. [2]) both uses the density `nHats(iSpeciesB)` while the other contributions to the total collision operator `CHat` uses `nHats(iSpeciesA)` instead.

Rather than creating many if-statements in the current implementation of the collision operator, for sake of clarity we separate the two implementations with one if-statement directly before the various contributions to the collision operator are calculated. If `poloidalVariationInCollisionOperator = .false.` the first block is executed, which contains the original code.

```
if (.not. poloidalVariationInCollisionOperator) then !! Added by AI (2017-09).
```

Otherwise, if `poloidalVariationInCollisionOperator = .true.`, `includePhi1 = .true.` and `includePhi1InKineticEquation = .true.` we proceed to the new block where poloidal density variation is included.

```
else if (includePhi1 .and. includePhi1InKineticEquation) then! Do with
    poloidal density variation
```

Since we now will have a `Phi1Hat` dependence in the various terms of the collision operator we will have to iterate over `itheta` and `izeta` when calculating the various contributions. Before calculating each term, we define a pre-factor (`PreFactor`) as is specified in Eq. (3), taking the correct species index into account.

```
do itheta=ithetaMin,ithetaMax
  do izeta=izetaMin,izetaMax
    ! Generate preFactor for nHats(iSpeciesA) terms
    preFactor =
      exp(-Zs(iSpeciesA)*alpha*Phi1Hat(itheta,izeta)/Thats(iSpeciesA))
```

Because of the `itheta`, `izeta` dependence we have redefined `CECD` and `nuDHat` to their equivalents `CECDpol` and `nuDHatpol` with space for their `itheta` and `izeta` components.

The rest of this block is the same as in the original code, except that we do the second iteration over `itheta`, `izeta` slightly earlier, and include the appropriate `PreFactor` in front of additional contributions to the total collision operator `CHat`.

Temperature equilibration term

In the original version of the code, the temperature equilibration term is calculated in `evaluateResidual.F90` by using `whichMatrix = 2` when calling `PopulateMatrix.F90`. The output is then multiplied with the first order distribution function f_{0s} . Since the original version of `SFINCS` already includes Φ_1 in the definition of f_{0s} (in the subroutine `init_f0()` inside `PopulateMatrix.F90`), the same routines as were presented above can be used also if `includeTemperatureEquilibrationTerm = .true..`

Jacobian

If the modifications explained in the previous section are implemented, the $\partial C / \partial f_{1s}$ contribution to the Jacobian is at this point already taking the density variations into account. We need to add the calculation of $\partial C / \partial \Phi_1$.

The $\partial C / \partial \Phi_1$ term

We begin with treating the other two terms appearing in Eq. (4) and will then look at the temperature equilibration term.

To implement the derivative with respect to Φ_1 we need to redo the calculation of the various contributions to the collision operator, but including a new pre-factor, `PreFactorJ` as specified in Eq. (4). This should only be done when the Jacobian (`whichmatrix = 1`) or the preconditioner (`whichmatrix = 0`) is calculated.

```
if (whichMatrix == 1 .or. whichMatrix == 0) then
  ! Generate preFactorJ for nHats(iSpeciesB) terms
  preFactorJ = (-Zs(iSpeciesA)*alpha/Thats(iSpeciesA)) &
    *exp(-Zs(iSpeciesA)*alpha*Phi1Hat(itheta,izeta)/Thats(iSpeciesA))
```

The result is saved in the new matrices nuDHatpolJ, CECDpolJ etc. Same calculations are then repeated for these terms (except for the different pre-factor), and the result is saved in CHatJ.

In contrary to the residual (and the $\partial C/\partial f_{1s}$ terms), when calculating $\partial C/\partial \Phi_1$ we have to include the first order distribution function before saving the result into the main matrix. For this purpose we first have to initiate this distribution function, by reading the appropriate terms in the current state vector.

```
do ix= max(ixMinCol,min_x_for_L(L)),Nx
  ! Generate f1b from state vector
  index = getIndex(iSpeciesB,ix,L+1,itheta,izeta,BLOCK_F)
  f1b(ix) = stateArray(index + 1)
```

We multiply the terms in the collision operator with this distribution function.

```
CHatTimesf = matmul(CHatJ,f1b)
```

The result is then saved into the $\partial \hat{C}/\partial \hat{\Phi}_1$ block of the main matrix,

```
do ix_row=max(ixMin,min_x_for_L(L)),Nx
  rowIndex=getIndex(iSpeciesA,ix_row,L+1,itheta,izeta,BLOCK_F)
  ! Get column index for the d/dPhi1 terms
  colIndex=getIndex(1,1,1,itheta,izeta,BLOCK_QN)
  ! Save into the main matrix, note that here we only use ix_row since
    CHatTimesf is now a vector
  call MatSetValue(matrix, rowIndex, colIndex, &
    -nu_n*CHatTimesf(ix_row), ADD_VALUES, ierr)
  ! need to use MatSetValue, otherwise petsc gives error
end do ! ix_row
```

Temperature equilibration

If the temperature equilibration terms should be included, we have to add additional terms into the main matrix, when calculating $\partial \hat{C}/\partial \hat{\Phi}_1$. These terms are different from the other contributions since now we need to multiply the output with $f_{0s} = f_{Ms}(\psi)e^{-Z_s e \Phi_1(\theta, \zeta)/T_s}$ (instead of the first order distribution function f_{1s}). Because of this, we need to generate the Maxwellian distribution function for iSpeciesB.

```
if (includeTemperatureEquilibrationTerm .and. L==0) then
  fM(ix) =
    sqrt(mhats(iSpeciesB)/Thats(iSpeciesB))*mhats(iSpeciesB)/Thats(iSpeciesB)&
    * nhats(iSpeciesB)/(pi*sqrt(pi)*expnb2(ix)
end if
```

Because of the Phi1Hat dependence in the zeroth order distribution function we get a different contribution to the Jacobian, as specified in the first term of Eq. (4). In the code, we need to use both the residual term CHat and the Jacobian terms CHatJ to obtain the correct result.

```
CHatTimesf = matmul(CHatJ + CHat*(-Zs(iSpeciesB)*alpha/Thats(iSpeciesB))&
    *exp(-Zs(iSpeciesB)*alpha*Phi1Hat(itheta,izeta)/Thats(iSpeciesB)),fM)
```

What is left is to add this term into the main matrix, using the the same procedure as was explained in the previous section.

Pure pitch angle scattering operator

The implementation of poloidal density variation in the case the pure pitch angle scattering collision operator is used (corresponding to the case `collisionoperator = 1`), is slightly easier because of the existing loop over `itheta`, `izeta` and since we are working with a scalar `CHat_element`. Also, since $L > 0$ always for this term, no temperature equilibration has to be included. The only block we need to change is when we save the collision operator into the main matrix.

We introduce the factor `preFactor` which is in normal cases equal to one, but when `poloidalVariationInCollisionOperator = .true.` it is equal to the exponential pre-factor containing Φ_1 in Eq. (3).

Before saving into the main matrix, we multiply the collision operator with this factor.

```
call MatSetValue(matrix, index, colIndex, &
    -nu_n*CHat_element*preFactor, ADD_VALUES, ierr)
```

Jacobian, the $\partial C / \partial \Phi_1$ term

Just as before, the $\partial C / \partial f_1$ terms are the same as in the residual. What is different is the $\partial C / \partial \Phi_1$ term. The procedure to calculate this term is however very similar, to the case with the residual. The only difference is the different pre-factor which now also is multiplied with the distribution function from the state vector.

```
preFactor = -Zs(iSpeciesA)*alpha/Thats(iSpeciesA)*exp(-Zs(iSpeciesA) &
    *alpha*Phi1Hat(itheta,izeta)/Thats(iSpeciesA))*stateArray(index + 1)
```

Current state of the project

Presently, the modifications discussed in the previous section have been implemented into the “`poloidalVariationInCollisionOperator`” branch of **SFINCS**. What remains is to benchmark the output with theoretical predictions, using for example the model described in Ref. [3].

Using the new implementation, for “reasonable” input values of impurity charge and main ion temperature gradient, no particular poloidal asymmetries have been observed, and the result is fairly similar if the original code is used instead. Attempts to benchmark with the model described in Ref. [3] suggests that much larger ion temperature gradients and impurity charge has to be used, in order to observe poloidal asymmetries in the perturbed density. The problem is, at this high gradients and impurity charge, it becomes

difficult to converge. In general, it seems that a slightly higher `Ntheta` has to be used, when including poloidal variations in the collision operator.

References

- [1] S. Buller, *Poloidal variation in collision operator* (2017).
- [2] M. Landreman, *Technical Documentation for version 3 of SFINCS* (2014).
- [3] T. Fülöp, P. Helander *Phys. Plasmas* **6**, 3066 (1999)