```fortran
subroutine jacobian(A,u,J,beta,lambda)

   use header


   implicit none
   ! Evaluate the Jacobian F'(U)

   type(matrix), intent(in) :: A
   type(matrix), intent(out) :: J
   type(vector), intent(in) :: u
   integer :: i                  !,nrows
   real(kind=8) :: y, beta, lambda

   ! Initialise the Jacobian with the linear part A

   ! Note: As m grows 'J=A' becomes increasingly inefficient. using dcopy
   ! from LAPACK to efficiently copy the elements required would be better.
   ! Unfortuately, I was not able to get this idea working!
   ! Something along the lines of this code below was what I was thinking!

   !call dcopy(A%nnz, A%aa(A%ii(A%ibeg):(A%ii(A%iend+1)-1)),1,J%aa(J%ii(J%ibeg):(J%ii(J%i
end+1)-1)),1)
   !call dcopy(A%n+1, A%ii(1:(A%n+1)), 1, J%ii(1:(J%n+1)),1)
   !call dcopy(A%nnz, A%jj(A%ii(A%ibeg):(A%ii(A%iend+1)-1)),1,J%jj(J%ii(J%ibeg):(J%ii(J%i
end+1)-1)),1)

   ! Inefficient solution but works!

   J = A

   ! Calculates F'(U) = A - diag(G'(U))
   do i=A%ibeg,A%iend
      y = 1.0_8 + beta*u%xx(i)
      J%aa(J%ii(i)) = J%aa(j%ii(i)) - lambda*exp(u%xx(i)/y)/(y*y)
   end do

end subroutine jacobian
```