```
subroutine evalFuncGrad(this, num_dv, x, q, f, dfdx)
  class(solver)                                :: this
  type(scalar), dimension(:), intent(in)   :: x, q
  type(scalar), dimension(:), intent(inout):: dfdx
  type(scalar), intent(out)                    :: f
  ...

  ! Solve the ODE/PDE
  call this % solvePDE(q, x)

  ! Evaluate the function
  call this % evalFunc(q, x, f)

  vars: do m = 1, num_dv
     ! Store the original x(m) value
     xtmp = x(m)

     ! Perturb the m-th index of x
#if defined USE_COMPLEX
     x(m) = cmplx(dble(x(m)), 1.0d-16)
#else
     x(m) = x(m) + dh
#endif
     ! Solve the ODE/PDE
     call this % solvePDE(q, x)

     ! Evaluate the function
     call this % evalFunc(q, x, ftmp)

     ! Restore x
     x(m) = xtmp

     ! Find the FD/CSD derivative
#if defined USE_COMPLEX
     dfdx(m) = aimag(ftmp)/1.0d-16
#else
     dfdx(m) = (ftmp-f)/dh
#endif
  end do vars

end subroutine evalFuncGrad
```