Note on vector potential for a solenoid

Juan C. Gallardo[†]

Brookhaven National Laboratory

Abstract

We write approximate expressions for the vector potential of a solenoid modeled as a collection of coils and as thick cylindrical current sheets.

August 20, 1999

I INTRODUCTION

The vector potential is an ubiquitous quantity in accelerator physics. Hamiltonian formalism require it for a consistent and correct description of particle dynamics. Following the history of normalized emittance in a magnetic channel, also, make it necessary to know the vector potential. The canonical momentum is defined as,

$$\vec{P} = \vec{p} + q\vec{A} \tag{1}$$

where $\vec{p}$ is the kinetic momentum, q is the charge and $\vec{A}$ is the vector potential.

II CALCULATIONS

The vector potential for a single planar coil on axis at position S and radius R, is given by[1],[2]

$$A_\phi(r,s;R,S) = \begin{cases} \frac{\mu_o IR}{\pi}\int_0^\infty dk \cos k(s-S) K_1(kR) I_1(kr) & r < R \\ \frac{\mu_o IR}{\pi}\int_0^\infty dk \cos k(s-S) I_1(kR) K_1(kr) & r > R \end{cases} \tag{2}$$

[†] gallardo@bnl.gov; http://pubweb.bnl.gov/people/gallardo/errors/

where $J_1(z), I_1(z)$ and $K_1(z)$ are Bessel functions of first kind and modified Bessel respectively.

We notice that powers of k could be obtained by taking the derivatives respect to s of the function $\sin k(s-S)$. Expressing the Bessel function as a series,

$$I_1(kr) = (\frac{1}{2}kr)\sum_{m=0}^{\infty}(\frac{1}{4}r^2)^m \frac{k^{2m}}{m!\Gamma(m+2)} \tag{3}$$

then the vector potential can be written as

$$A_\phi = \frac{\mu_o IR}{\pi}\int_0^\infty dk \cos k(s-S) K_1(kR)\frac{1}{2}kr\left\{1 + \frac{1}{4}\frac{r^2k^2}{\Gamma(3)} + ...\right\} \tag{4}$$

Notice that the lowest order contains $k\cos k(s-S)$; the next $k^3\cos k(s-S)$ and so on ...; now,

$$\begin{aligned} k\cos k(s-S) &= \frac{\partial}{\partial s}\sin k(s-S) \\ k^3\cos k(s-S) &= \frac{\partial^3}{\partial s^3}\sin k(s-S) \\ k^5\cos k(s-S) &= \frac{\partial^5}{\partial s^5}\sin k(s-S) \\ ... &= ... \end{aligned} \tag{5}$$

Substituting in Eq.3 we obtain,

$$A_\phi(r,s;R,S) = \frac{\mu_o IR}{\pi}\int_0^\infty dk K_1(kR)\frac{1}{2}r\frac{\partial}{\partial s}\sum_{m=0}^{\infty}\frac{(-)^m(\frac{1}{4}r^2)^m}{m!\Gamma(m+2)}\frac{\partial^{2m}}{\partial s^{2m}}\sin k(s-S) \tag{6}$$

Next we evaluate the integral over k [3]

$$\int_0^\infty dk K_1(kR)\sin k(s-S) = \frac{\pi}{4}\frac{2(s-S)}{R\sqrt{R^2+(s-S)^2}} \tag{7}$$

and define

$$A_0(s;R,S) = \frac{\mu_o I}{2}\frac{(s-S)}{\sqrt{R^2+(s-S)^2}} \tag{8}$$

This approach is elegant because all dependence on k is restricted to $K_1(kR)$ and $\sin k(s-S)$ which has been integrated, see Eq. 7 and the r dependence is a power series expansion with coefficients, functions of s, obtained as derivatives respect to s of the vector potential on axis, see Eq. 8.

A Single planar coil

The vector potential is a power series expansion in r of the form,

$$A_\phi(r,s;R,S) = \frac{\mu_o I}{2}\left\{\frac{1}{2}rA_1 - \frac{1}{16}r^3A_3 + \frac{1}{384}r^5A_5 - \frac{1}{18432}r^7A_7 + ...\right\} \tag{9}$$

with coefficients

$$A_i == \frac{\partial^i}{\partial s^i}\frac{(s-S)}{\sqrt{R^2+(s-S)^2}} \tag{10}$$

These coefficients are trivially found using *Mathematica*[4]. For practical use we have written a fortran subroutine, which is available at the WEB site

`http://pubweb.bnl.gov/people/gallardo/errors/`.

B Solenoid modeled as a thick cylindrical current sheet

First we integrate Eq. 8 over S and R obtaining [5]

$$A_0(s;L,R_2,R_1) = \frac{-\mu_o I}{8L(R_2-R_1)}\left\{(s-S)^2\ln\left[R+\sqrt{R^2+(s-S)^2}\right] + R\sqrt{R^2+(s-S)^2}\right\}\Big|_{-L}^{L}\Big|_{R_1}^{R_2} \tag{11}$$

As before,

$$A_\phi(r,s) = \frac{-\mu_o I}{2}\left\{\frac{1}{2}rC_1 - \frac{1}{16}r^3C_3 + \frac{1}{384}r^5C_5 - \frac{1}{18432}r^7C_7 + ...\right\}\Big|_{-L}^{L}\Big|_{R_1}^{R_2} \tag{12}$$

with coefficients

$$C_i = \frac{\partial^i}{\partial s^i}\left((s-S)^2\ln\left[R+\sqrt{R^2+(s-S)^2}\right] + R\sqrt{R^2+(s-S)^2}\right) \tag{13}$$

III SUMMARY

We have written approximate expressions for the vector potential of a solenoid modeled as a collection of coils and as thick cylindrical current sheets. A simple fortran code give the coefficients of the expansion.

ACKNOWLEDGMENTS

We thank K. McDonald for and S. Kahn for discussions and suggestions.

REFERENCES

[1] J. Gallardo, *Errors in a discrete solenoid* (1999), BNL-CAP-Tech. Notes MU-031, http://www-mucool.fnal.gov/mcnotes/muc0037.ps.

[2] W. Panofsky and M. Phillips, *Classical Electricity and Magnetism* (Addison-Wesley, New York, 1965).

[3] I. Gradshteyn and I. Ryzhik, *Tables of integrals, Series and Products, pp. 712* (Academic Press, New York, 1980).

[4] S. Wolfram, *Mathematica, A System for Doing Mathematics by Computer, 2nd edition* (Addison-Wesley, Redwood City, CA, 1991).

[5] J. Gallardo, *Errors in a continuos solenoid* (1999), BNL-CAP-Tech. Notes MU-xxx, http://www-mucool.fnal.gov/mcnotes/muc0041.ps.

A A simple fortran code: vector_pot_coil.f

```
      program vector_potential_coil ! vector potential for single coil.
      implicit double precision (a-h,o-z)
c
      dimension vector(200,500)
      dimension rad(200),xlong(500)
     common/parameters/XL,R1,R2
c   eta is position of center and rho radius
      external f1,f2,f3,f4,f5
```

```
      open(unit=29,file='pot_sol_coil.dat',status='unknown',
     +  form='unformatted')
        XL=0.2  ! total length is 2*L=0.4 m
        R1=0.7
        R2=R1+0.15
        n_radial_points=200
        n_long_points=500
        ds=60*XL/n_long_points
        dr=R1/n_radial_points   ! inside solenoid
        eta=0    ! long position of the coil
c potential as a power series expansion in r
c
c        b00=mu_o*I/(R2-R1)*Ln(R2+sqrt(R2**2+XL**2))-Ln(R1+sqrt(R1**2+XL**2))
c        b00=log((R2+sqrt(R2**2+XL**2))/(R1+sqrt(R1**2+XL**2)))/(2*XL)
c        b00=mu_o*I/2/R
        b00=R1
c order of the expansion
c        type*,'up to which order in r? (1,3,5,7...(le.7))'
c        read*,iorder
        do k=1,n_radial_points
           do i=1,n_long_points
              vector(k,i)=0
              end do
              end do
c  loop over positions
            r=0
              do k=1,n_radial_points
```



```
           rad(k)=r
           s=-30*XL
           do i=1,n_long_points
          xlong(i)=s
           vec1=0.5*r*a1(s,eta)
c        do j=1,iorder,2
c           if(j.eq.3)vec2=-1/16.*r**3*a3(s,eta)
           vec3=-1/16.*r**3*a3(s,eta)
c           if(j.eq.5)vec3=1/384.*r**5*a5(s,eta)
           vec5=1/384.*r**5*a5(s,eta)
c           if(j.eq.7)vec4=-1/18432*r**7*a7(s,eta)
           vec7=-1/18432*r**7*a7(s,eta)
c           if(j.gt.7)type*,'order too large(max. is 7)'
c           end do
           vector(k,i)=b00*(vec1+vec3+vec5+vec7)
           if(i.eq.250)then
           write(20,*)rad(k),vector(k,i)
           endif
          s=s+ds
c        write(6,*)'r,s',r,s
        end do
        if(k.eq.100)then
           do i=1,n_long_points
           write(10,*)xlong(i),vector(k,i)
           end do
           endif
        r=r+dr
```

```
        end do
        write(29)n_radial_points,n_long_points
       write(29)vector
       write(6,*)'pot_sol_coil.dat contains:A(i,j)',
     +      n_radial_points,n_long_points
       write(6,*)'fort.10 contains: s, vector(100,s)'
       write(6,*)'fort.20 contains: r, vector(r,250)'
c 50    format(1x,6(g12.5,1x))
        stop
        end
c   FUNCTIONS
      real*8 function a1(ss,eta) ! first order
      implicit double precision (a-h,o-z)
      common/parameters/XL,R1,R2
c   eta is position of center and R1 radius
      aux0=sqrt(R1**2+(ss-eta)**2)
      aux1=1/aux0
      aux2=-(ss-eta)**2/aux0**3
      a1=aux1+aux2
      end
c


      real*8 function a3(ss,eta) ! 3rd order
      implicit double precision (a-h,o-z)
c   eta is position of center and R1 radius
      common/parameters/XL,R1,R2
      aux0=sqrt(R1**2+(ss-eta)**2)
```



```
      aux1=-15*(ss-eta)**4/aux0**7
      aux2=18*(ss-eta)**2/aux0**5
      aux3=-3/aux0**3
      a3=aux1+aux2+aux3
      end
c
      real*8 function a5(ss,eta) ! 5th order
      implicit double precision (a-h,o-z)
c   eta is position of center and R1 radius
      common/parameters/XL,R1,R2
      aux0=sqrt(R1**2+(ss-eta)**2)
      aux1=-945*(ss-eta)**6/aux0**11
      aux2=1575*(ss-eta)**4/aux0**9
      aux3=-675*(ss-eta)**2/aux0**7
      aux4=45/aux0**5
      a5=aux1+aux2+aux3+aux4
      end
c
      real*8 function a7(ss,eta) ! 7th order
      implicit double precision (a-h,o-z)
c  eta is position of center and R1 radius
      common/parameters/XL,R1,R2
      aux0=sqrt(R1**2+(ss-eta)**2)
      aux1=-135135*(ss-eta)**8/aux0**15
      aux2=291060*(ss-eta)**6/aux0**13
      aux3=198450*(ss-eta)**4/aux0**11
      aux4=44100*(ss-eta)**2/aux0**9
```

```
      aux5=1575/aux0**7
      a5=aux1+aux2+aux3+aux4+aux5

      end
```

B A simple fortran code: vector_pot_sheet.f

```
        program vector_potential_sheet ! vector potential for thick
c                                        cylindrical current sheet.
        implicit double precision (a-h,o-z)
c
        dimension vector(200,500)
        dimension rad(200),xlong(500)
      common/parameters/XL,R1,R2
c   R1 internal radius; R2 external radius
        external a1,a3,a5,a7
       open(unit=29,file='pot_sol_sheet.dat',status='unknown',
     +  form='unformatted')
        XL=0.2  ! total length is 2*L=0.4 m
        R1=0.7
        R2=R1+0.15
        n_radial_points=200
        n_long_points=500
        ds=60*XL/n_long_points
        dr=R1/n_radial_points   ! inside solenoid
        eta=0    ! centered at origen
```



```
c potential as a power series expansion in r
c
c        b00=mu_o*I/(R2-R1)*Ln(R2+sqrt(R2**2+XL**2))-Ln(R1+sqrt(R1**2+XL**2))
        b00=log((R2+sqrt(R2**2+XL**2))/(R1+sqrt(R1**2+XL**2)))/(2*XL)
c both expression above are for cylindrical current sheet
c        b00=mu_o*I/2/R  ! for coil
c        b00=R1    ! for coil
c order of the expansion
c        type*,'up to which order in r? (1,3,5,7...(le.7))'
c        read*,iorder
        do k=1,n_radial_points
           do i=1,n_long_points
              vector(k,i)=0
              end do
              end do
c  loop over positions
            r=0
              do k=1,n_radial_points
           rad(k)=r
           s=-30*XL
           do i=1,n_long_points
          xlong(i)=s
           vec1=0.5*r*(a1(s,R2,Xl)-a1(s,R2,-XL)-
     +          (a1(s,R1,XL)-a1(s,R1,-XL)))
c        do j=1,iorder,2
c           if(j.eq.3)vec2=-1/16.*r**3*a3(s,eta)
           vec3=-1/16.*r**3*(a3(s,R2,Xl)-a3(s,R2,-XL)-
```

```
     +          (a3(s,R1,XL)-a3(s,R1,-XL)))
c           if(j.eq.5)vec3=1/384.*r**5*a5(s,eta)
           vec5=1/384.*r**5*(a5(s,R2,Xl)-a5(s,R2,-XL)-
     +          (a5(s,R1,XL)-a5(s,R1,-XL)))
c           if(j.eq.7)vec4=-1/18432*r**7*a7(s,eta)
           vec7=-1/18432*r**7*(a7(s,R2,Xl)-a7(s,R2,-XL)-
     +          (a7(s,R1,XL)-a7(s,R1,-XL)))
c           if(j.gt.7)type*,'order too large(max. is 7)'
c           end do
           vector(k,i)=-b00*(vec1+vec3+vec5+vec7) ! minus sign is arbitrary
           if(i.eq.250)then
           write(40,*)rad(k),vector(k,i)
           endif
          s=s+ds
c        write(6,*)'r,s',r,s
        end do
        if(k.eq.100)then
           do i=1,n_long_points
           write(30,*)xlong(i),vector(k,i)
           end do
           endif
        r=r+dr
        end do
        write(29)n_radial_points,n_long_points
       write(29)vector
       write(6,*)'pot_sol_sheet.dat contains:A(i,j)',
     +      n_radial_points,n_long_points
```



```
       write(6,*)'fort.30 contains: s, vector(100,s)'
       write(6,*)'fort.40 contains: r, vector(r,250)'
c 50    format(1x,6(g12.5,1x))
        stop
        end
c   FUNCTIONS
      real*8 function a1(ss,R,S) ! first order
      implicit double precision (a-h,o-z)
      common/parameters/XL,R1,R2
c
      aux0=(R*(ss - S))/Sqrt(R**2 + (ss - S)**2) +
     -  (ss - S)**3/
     -   ((R + Sqrt(R**2 + (ss - S)**2))*Sqrt(R**2 + (ss - S)**2))
     -    + 2*(ss - S)*Log(R + Sqrt(R**2 + (ss - S)**2))
      a1=aux0
      end
c


      real*8 function a3(ss,R,S) ! 3rd order
      implicit double precision (a-h,o-z)
      common/parameters/XL,R1,R2
c
      aux0=(-3*R*(ss - S))/(R**2 + (ss - S)**2)**1.5 +
     -  (12*(ss - S))/
     -   ((R + Sqrt(R**2 + (ss - S)**2))*Sqrt(R**2 + (ss - S)**2))
     -    + (3*R*(ss - S)**3)/(R**2 + (ss - S)**2)**2.5 -
     -  (9*(ss - S)**3)/
```

```
     -   ((R + Sqrt(R**2 + (ss - S)**2))*
     -     (R**2 + (ss - S)**2)**1.5) -
     -  (9*(ss - S)**3)/
     -   ((R + Sqrt(R**2 + (ss - S)**2))**2*(R**2 + (ss - S)**2))
     -   + (3*(ss - S)**5)/
     -   ((R + Sqrt(R**2 + (ss - S)**2))*
     -     (R**2 + (ss - S)**2)**2.5) +
     -  (3*(ss - S)**5)/
     -   ((R + Sqrt(R**2 + (ss - S)**2))**2*
     -     (R**2 + (ss - S)**2)**2) +
     -  (2*(ss - S)**5)/
     -   ((R + Sqrt(R**2 + (ss - S)**2))**3*
     -     (R**2 + (ss - S)**2)**1.5)
      a3=aux0
      end
c
      real*8 function a5(ss,R,s) ! 5th order
      implicit double precision (a-h,o-z)
c  eta is position of center and R1 radius
      common/parameters/XL,R1,R2
      aux0=(105*r*(-s + ss)**5)/(r**2 + (-s + ss)**2)**4.5 -
     -  (150*r*(-s + ss)**3)/(r**2 + (-s + ss)**2)**3.5 +
     -  (45*r*(-s + ss))/(r**2 + (-s + ss)**2)**2.5 +
     -  (24*(-s + ss)**7)/
     -   ((r**2 + (-s + ss)**2)**2.5*
     -     (r + Sqrt(r**2 + (-s + ss)**2))**5) +
     -  (60*(-s + ss)**7)/
```



```
     -   ((r**2 + (-s + ss)**2)**3*
     -     (r + Sqrt(r**2 + (-s + ss)**2))**4) -
     -  (120*(-s + ss)**5)/
     -   ((r**2 + (-s + ss)**2)**2*
     -     (r + Sqrt(r**2 + (-s + ss)**2))**4) +
     -  (90*(-s + ss)**7)/
     -   ((r**2 + (-s + ss)**2)**3.5*
     -     (r + Sqrt(r**2 + (-s + ss)**2))**3) -
     -  (240*(-s + ss)**5)/
     -   ((r**2 + (-s + ss)**2)**2.5*
     -     (r + Sqrt(r**2 + (-s + ss)**2))**3) +
     -  (190*(-s + ss)**3)/
     -   ((r**2 + (-s + ss)**2)**1.5*
     -     (r + Sqrt(r**2 + (-s + ss)**2))**3) +
     -  (105*(-s + ss)**7)/
     -   ((r**2 + (-s + ss)**2)**4*
     -     (r + Sqrt(r**2 + (-s + ss)**2))**2) -
     -  (300*(-s + ss)**5)/
     -   ((r**2 + (-s + ss)**2)**3*
     -     (r + Sqrt(r**2 + (-s + ss)**2))**2) +
     -  (285*(-s + ss)**3)/
     -   ((r**2 + (-s + ss)**2)**2*
     -     (r + Sqrt(r**2 + (-s + ss)**2))**2) -
     -  (90*(-s + ss))/
     -   ((r**2 + (-s + ss)**2)*
     -     (r + Sqrt(r**2 + (-s + ss)**2))**2) +
     -  (105*(-s + ss)**7)/
```

```
-   ((r**2 + (-s + ss)**2)**4.5*
-      (r + Sqrt(r**2 + (-s + ss)**2))) -
-  (300*(-s + ss)**5)/
-   ((r**2 + (-s + ss)**2)**3.5*
-      (r + Sqrt(r**2 + (-s + ss)**2))) +
-  (285*(-s + ss)**3)/
-   ((r**2 + (-s + ss)**2)**2.5*
-      (r + Sqrt(r**2 + (-s + ss)**2))) -
-  (90*(-s + ss))/
-   ((r**2 + (-s + ss)**2)**1.5*
-      (r + Sqrt(r**2 + (-s + ss)**2)))

      a5=aux0
      end
c
       real*8 function a7(ss,R,s)   ! 7th order
       implicit double precision (a-h,o-z)
       common/parameters/XL,R1,R2
       aux0=(10395*r*(-s + ss)**7)/(r**2 + (-s + ss)**2)**6.5 -
-  (19845*r*(-s + ss)**5)/(r**2 + (-s + ss)**2)**5.5 +
-  (11025*r*(-s + ss)**3)/(r**2 + (-s + ss)**2)**4.5 -
-  (1575*r*(-s + ss))/(r**2 + (-s + ss)**2)**3.5 +
-  (720*(-s + ss)**9)/
-   ((r**2 + (-s + ss)**2)**3.5*
-      (r + Sqrt(r**2 + (-s + ss)**2))**7) +
-  (2520*(-s + ss)**9)/
-   ((r**2 + (-s + ss)**2)**4*
```



```
-      (r + Sqrt(r**2 + (-s + ss)**2))**6) -
-  (4200*(-s + ss)**7)/
-   ((r**2 + (-s + ss)**2)**3*
-      (r + Sqrt(r**2 + (-s + ss)**2))**6) +
-  (5040*(-s + ss)**9)/
-   ((r**2 + (-s + ss)**2)**4.5*
-      (r + Sqrt(r**2 + (-s + ss)**2))**5) -
-  (12600*(-s + ss)**7)/
-   ((r**2 + (-s + ss)**2)**3.5*
-      (r + Sqrt(r**2 + (-s + ss)**2))**5) +
-  (8568*(-s + ss)**5)/
-   ((r**2 + (-s + ss)**2)**2.5*
-      (r + Sqrt(r**2 + (-s + ss)**2))**5) +
-  (7560*(-s + ss)**9)/
-   ((r**2 + (-s + ss)**2)**5*
-      (r + Sqrt(r**2 + (-s + ss)**2))**4) -
-  (22050*(-s + ss)**7)/
-   ((r**2 + (-s + ss)**2)**4*
-      (r + Sqrt(r**2 + (-s + ss)**2))**4) +
-  (21420*(-s + ss)**5)/
-   ((r**2 + (-s + ss)**2)**3*
-      (r + Sqrt(r**2 + (-s + ss)**2))**4) -
-  (6930*(-s + ss)**3)/
-   ((r**2 + (-s + ss)**2)**2*
-      (r + Sqrt(r**2 + (-s + ss)**2))**4) +
-  (9450*(-s + ss)**9)/
-   ((r**2 + (-s + ss)**2)**5.5*
```

```
-      (r + Sqrt(r**2 + (-s + ss)**2))**3) -
-   (29400*(-s + ss)**7)/
-    ((r**2 + (-s + ss)**2)**4.5*
-      (r + Sqrt(r**2 + (-s + ss)**2))**3) +
-   (32130*(-s + ss)**5)/
-    ((r**2 + (-s + ss)**2)**3.5*
-      (r + Sqrt(r**2 + (-s + ss)**2))**3) -
-   (13860*(-s + ss)**3)/
-    ((r**2 + (-s + ss)**2)**2.5*
-      (r + Sqrt(r**2 + (-s + ss)**2))**3) +
-   (1680*(-s + ss))/
-    ((r**2 + (-s + ss)**2)**1.5*
-      (r + Sqrt(r**2 + (-s + ss)**2))**3) +
-   (10395*(-s + ss)**9)/
-    ((r**2 + (-s + ss)**2)**6*
-      (r + Sqrt(r**2 + (-s + ss)**2))**2) -
-   (33075*(-s + ss)**7)/
-    ((r**2 + (-s + ss)**2)**5*
-      (r + Sqrt(r**2 + (-s + ss)**2))**2) +
-   (37485*(-s + ss)**5)/
-    ((r**2 + (-s + ss)**2)**4*
-      (r + Sqrt(r**2 + (-s + ss)**2))**2) -
-   (17325*(-s + ss)**3)/
-    ((r**2 + (-s + ss)**2)**3*
-      (r + Sqrt(r**2 + (-s + ss)**2))**2) +
-   (2520*(-s + ss))/
-    ((r**2 + (-s + ss)**2)**2*
```



```
-      (r + Sqrt(r**2 + (-s + ss)**2))**2) +
-   (10395*(-s + ss)**9)/
-    ((r**2 + (-s + ss)**2)**6.5*
-      (r + Sqrt(r**2 + (-s + ss)**2))) -
-   (33075*(-s + ss)**7)/
-    ((r**2 + (-s + ss)**2)**5.5*
-      (r + Sqrt(r**2 + (-s + ss)**2))) +
-   (37485*(-s + ss)**5)/
-    ((r**2 + (-s + ss)**2)**4.5*
-      (r + Sqrt(r**2 + (-s + ss)**2))) -
-   (17325*(-s + ss)**3)/
-    ((r**2 + (-s + ss)**2)**3.5*
-      (r + Sqrt(r**2 + (-s + ss)**2))) +
-   (2520*(-s + ss))/
-    ((r**2 + (-s + ss)**2)**2.5*
-      (r + Sqrt(r**2 + (-s + ss)**2)))
    a5=aux0
    end
```