Principe - Physics 265 PS7

January 6, 2024

1 Physics 265 Problem Set 7

Rene L. Principe Jr.
PhD Physics
2015-04622

[130]:
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from tqdm import tqdm
from matplotlib import cm
```

2 Problem 7.1

2.1 Primary Aberration of a Thin Lens (BaSF10)

(Section 5.6 Born & Wolf). Plot the Seidel aberration function (similar to Figure 5.3) that is due to a double convex thin lens which is made of your first dielectric material ($n_1 = = n_2 = n_{air} = 1$; $r_1 = 3$ cm, $r_2 = 4$ cm, A = 550 nm*, $b_1 = 0.8$, $b_2 = 0.9$) for the following cases (via Eqn 15): (a) Spherical aberration, (b) Coma, (c) Astigmatism, (d) Curvature of Field and (e) Total aberration.

`Section 5.6 Born & Wolf: Eq. 13`

$$B = h^4 U \tag{1}$$

$$F = h^4 k U + h^2 V \tag{2}$$

$$C = h^4 k^2 U + 2h^2 k V + \frac{1}{2} P \tag{3}$$

$$D = h^4 k^2 U + 2h^2 k V + \frac{n+1}{2n} P \tag{4}$$

$$E = h^4 k^3 U + 3h^2 k^2 V + k\frac{3n+1}{2n} P \tag{5}$$

Section 5.6 Born & Wolf: Eq. 14

where

$$U = \frac{1}{2}\beta + \frac{n_2}{8(n-1)^2}P^3 - \frac{n}{2(n+2)}K^2P + \frac{1}{2n(n+2)}P\left[\frac{n+2}{2(n-1)}\sigma + 2(n+1)K^2\right] \tag{6}$$

$$V = \frac{1}{2n}P\left[\frac{n+1}{2(n-1)}\sigma + (2n+1)K\right] \tag{7}$$

Setting h = -1 and k = 0, Eq. 13 reduces into

$$B = -U \qquad \text{(Spherical Aberration)} \tag{8}$$

$$C = \frac{1}{2}P \qquad \text{(Astigmatism)} \tag{9}$$

$$D = \frac{n+1}{2n}P \qquad \text{(Curvature of Field)} \tag{10}$$

$$E = 0 \qquad \text{(Distortion)} \tag{11}$$

$$F = V \qquad \text{(Coma)} \tag{12}$$

where

$$P = (n-1)\left(\frac{1}{r_1} - \frac{1}{r_2}\right) \tag{13}$$

$$\sigma = (n-1)\left(\frac{1}{r_1} + \frac{1}{r_2}\right) \tag{14}$$

$$\beta = (n-1)\left(\frac{b_1}{r_1^3} - \frac{b_2}{r_2^3}\right) \tag{15}$$

$$K = -\frac{1}{s_1} - \frac{P}{2} \tag{16}$$

[131]:
```
def B(n, beta_, K_, P_):
    term1 = beta_/2
    term2 = (P_**3 * n**2)/(8*(n-1)**2)
    term3 = K_**2 * P_ * n/(2*(n+2))

    if n==0:
        term4 = 0
```

```
    else :
        term4 = (P_/(2*n*(n+2)))*(K_*2*(n+1)*(n+2)/(2*(n-1)))**2
    return term1 + term2 - term3 + term4

def C(P_):
    return P_/2

def D(n,P_):
    if n == 0 :
        return 0
    else :
        return P_*(n+1)/(2*n)

def E(n):
    return 0

def F(n, P_, K_, sigma_):

    term1 = sigma_ *(n+1)/(2*(n-1))
    term2 = K_*(2*n+1)

    if n == 0:
        factor1 = 0
    else :
        factor1 = P_/(2*n)

    return factor1 * ( term1 + term2 )
```

[132]:
```
def P(n, r1, r2):
    return (n-1)*(1/r1 - 1/r2)

def sigma(n, r1, r2):
    return (n-1)*(1/r1 + 1/r2)

def beta(n, b1, b2, r1, r2):
    return (n-1)*((b1/r1**2) - b2/r2**2)

def K(s1, P_):
    return -1/s1 - P_/2
```

2.2 $\mathbf{n_1(\lambda)}$ of BaSF10

$$n_1^2(\lambda) = 2.6531250 - 8.1388553 \times 10^{-3}\lambda^2 + 2.2995643 \times 10^{-2}\lambda^{-2} + 7.3535957 \times 10^{-4}\lambda^{-4} \quad (17)$$

$$-1.3407390 \times 10^{-5}\lambda^{-6} + 3.6962325 \times 10^{-6}\lambda^{-8} \quad (18)$$

[133]:
```
def BaSF10_n(lambda_):
    term1 = 8.1388553*10**-3 * lambda_**2
    term2 =  2.2995643*10**-2 * lambda_**-2
    term3 =  7.3535957*10**-4 * lambda_**-4
    term4 =  1.3407390*10**-5 * lambda_**-6
    term5 =  3.6962325*10**-6 * lambda_**-8
    return np.sqrt(2.6531250 - term1 + term2 + term3 - term4 + term5)
```

[134]:
```
def rho(x,y):
    return np. sqrt(x**2 + y**2)
```

[135]:
```
r1 = 3
r2 = -4

n1 = 1.00
n3 = 1.00

b1 = 0.8
b2 = 0.9

s1 = np.infty
lambda_list = np.array([400, 600, 800])
```

[136]:
```
xx = np. linspace (-1, 1, 1000)
X, Y = np. meshgrid(xx, xx)
```

[137]:
```
abberations_list = []

for lambda_ in tqdm(lambda_list):

    n_ = BaSF10_n(lambda_/1e3)
    beta_ = beta(n_, b1, b2, r1, r2)
    sigma_ = sigma(n_, r1, r2)
    P_ = P(n_, r1 ,r2)
    K_ = K(s1, P_)

    B_ = -(1/4) * rho(X,Y)**4 * B(n_, beta_, K_, P_)
    C_ = -X**2 * C(P_)
    D_ = -(1/2) * rho(X,Y)**2 * D(n_, P_)
    E_ = X * E(n_)
    F_ = X * rho(X,Y)**2 * F(n_, P_, K_, sigma_)

    total = B_ + C_ + D_ + E_ + F_

    abberation_ = [B_, C_, D_, E_, F_, total]
    abberations_list.append(abberation_)
```

```
np.shape(abberations_list)
```

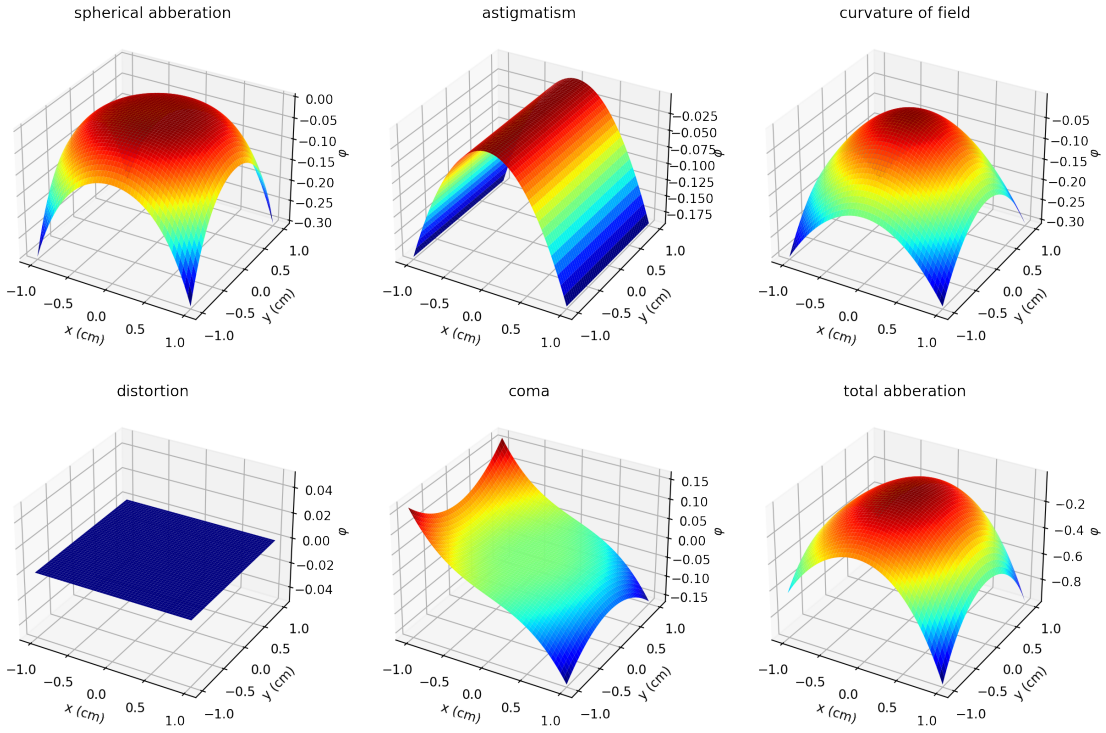
```
100%|                          | 3/3 [00:00<00:00,  6.05it/s]
```

[137]:
```
(3, 6, 1000, 1000)
```

[138]:
```
n_lambda = np.shape(abberations_list)[0]
n_abberations = np.shape(abberations_list)[1]

abberation_names = ['spherical abberation', 'astigmatism', 'curvature of field',
                    'distortion', 'coma', 'total abberation']

cmaps = ['winter', 'summer', 'autumn']
```

[139]:
```
i = 1

fig, axes = plt.subplots(figsize =(15,10), dpi = 200,
                         ncols=3, nrows = 2,
                         subplot_kw = dict ( projection = '3d'))

for j, ax in tqdm(enumerate(axes.ravel())):
    ax.plot_surface(X,Y, abberations_list[i][j], cmap = 'jet')
    ax.set_title(abberation_names[j])
    ax.set_xlabel('x (cm)')
    ax.set_ylabel('y (cm)')
    ax.set_zlabel('$\\varphi$')
plt.show()
```

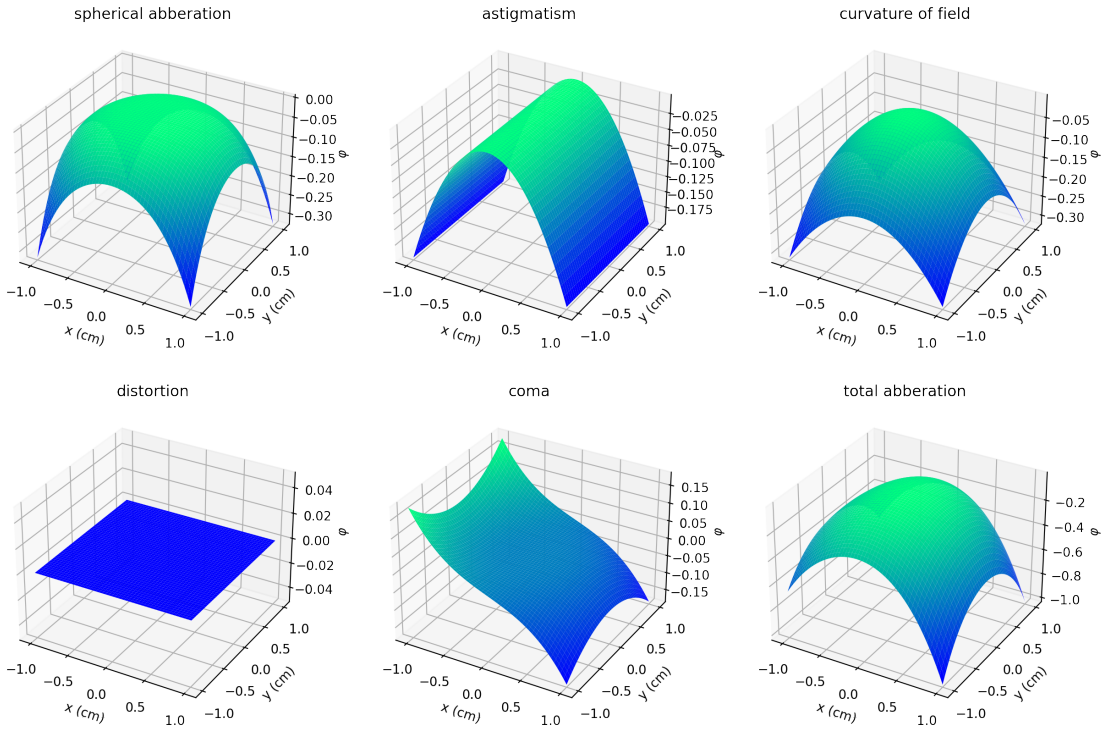
```
6it [00:00, 10.62it/s]
```



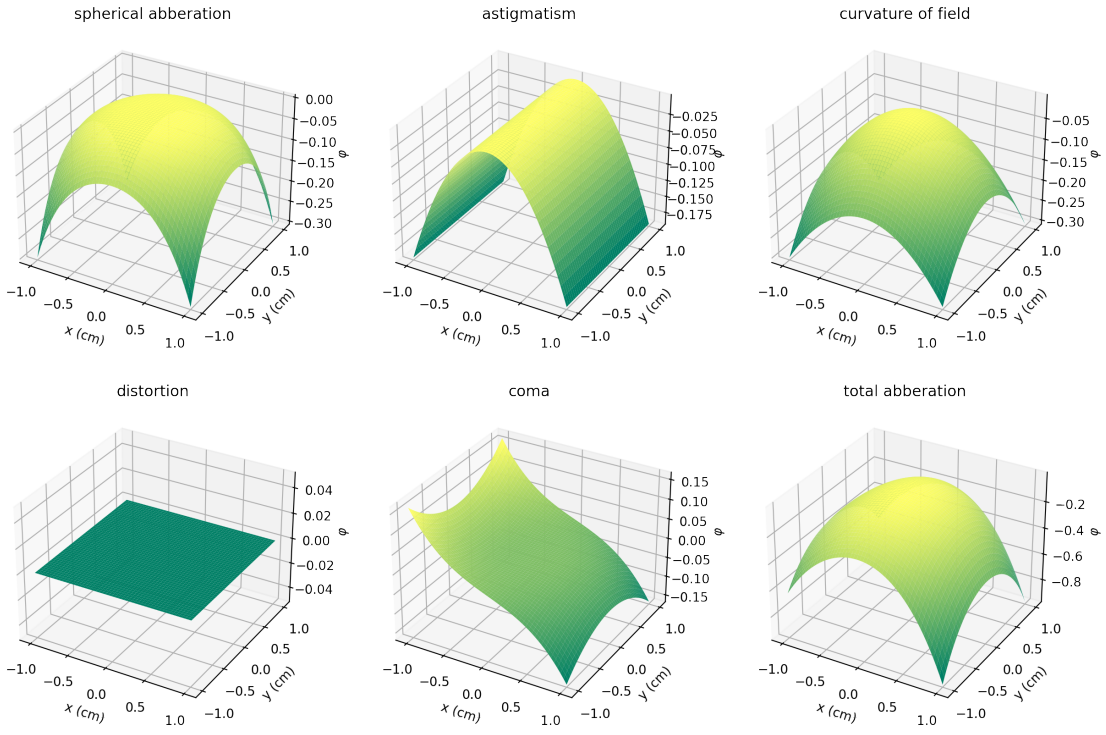
[140]:
```
for i in tqdm(range(n_lambda)):

    fig, axes = plt.subplots(figsize =(15,10), dpi = 200,
                             ncols=3, nrows = 2,
                             subplot_kw = dict ( projection = '3d'))

    for j, ax in enumerate(axes.ravel()):
        ax.plot_surface(X,Y, abberations_list[i][j], cmap = cmaps[i])
        ax.set_title(abberation_names[j])
        ax.set_xlabel('x (cm)')
        ax.set_ylabel('y (cm)')
        ax.set_zlabel('$\\varphi$')
    plt.suptitle('BaSF10: $\lambda$ = %.1f nm' % lambda_list[i], fontsize = 20)
    plt.show()
```

```
  0%|                                                    | 0/3 [00:00<?, ?it/s]
```

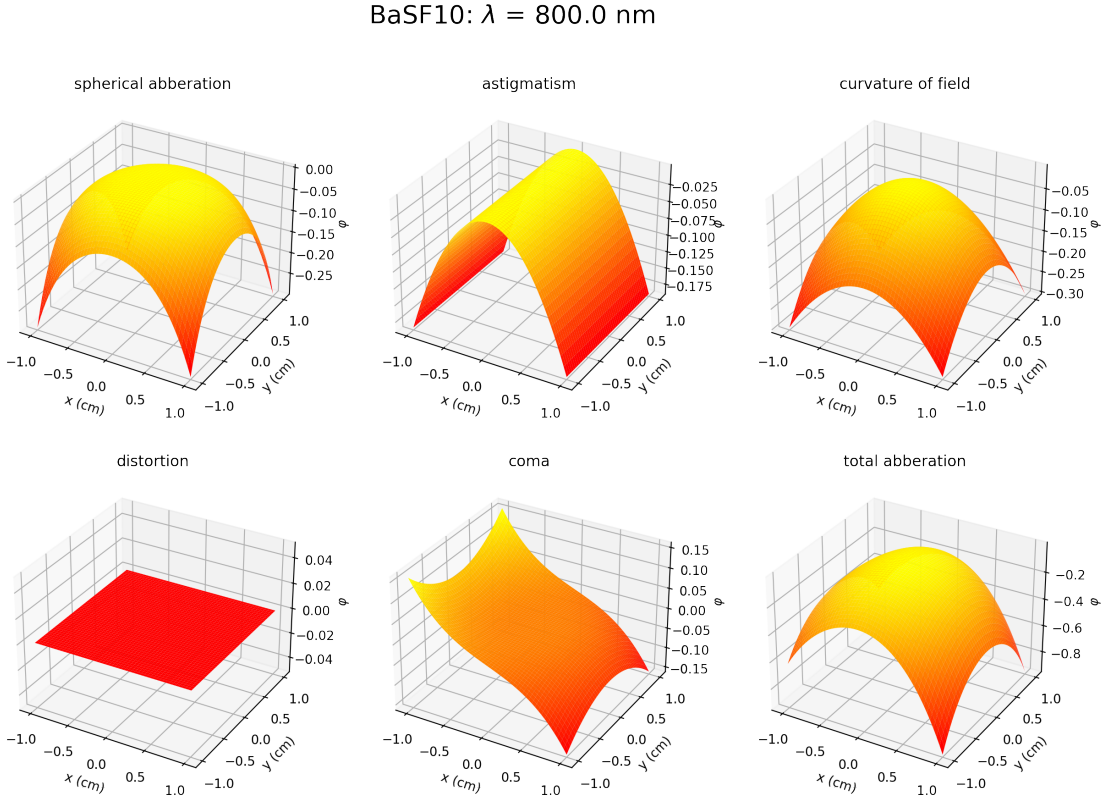
BaSF10: λ = 400.0 nm



33%| | 1/3 [00:01<00:03, 1.72s/it]



67%| | 2/3 [00:03<00:01, 1.60s/it]



100%| | 3/3 [00:05<00:00, 1.74s/it]

---

3 Problem 7.2

3.1 Primary Aberration of a Thin Lens (BaK1)

Plot the aberration function due to a thin lens that is made of your second dielectric material (n1 = = n2 = nair = 1; r1 = 3 cm, r2 = 4 cm, A = 550 nm, b1 = 0.9, b2 = 0.8) for the following cases (via Eqn 15): (a) Spherical aberration, (b) Coma, (c) Astigmatism, (d) Curvature of Field and (e) Total aberration.

3.2 $n_2(\lambda)$ of BaK1

$$n_2^2(\lambda) = 2.4333007 - 8.4931353 \times 10^{-3}\lambda^2 + 1.3893512 \times 10^{-2}\lambda^{-2} + 2.6798268 \times 10^{-4}\lambda^{-4} \quad (19)$$

$$-6.1946101 \times 10^{-6}\lambda^{-6} + 6.2209005 \times 10^{-7}\lambda^{-8} \quad (20)$$

[141]:
```
def BaK1_n(lambda_):
    term1 = 8.4931353*10**-3 * lambda_**2
    term2 =  1.3893512*10**-2 * lambda_**-2
```

```
    term3 =  2.6798268*10**-4 * lambda_**-4
    term4 =  6.1946101*10**-6 * lambda_**-6
    term5 =  6.2209005*10**-7 * lambda_**-8
    return np.sqrt(2.4333007 - term1 + term2 + term3 - term4 + term5)
```

[142]:
```
abberations_list_2 = []

for lambda_ in tqdm(lambda_list):

    n_ = BaK1_n(lambda_/1e3)
    beta_ = beta(n_, b1, b2, r1, r2)
    sigma_ = sigma(n_, r1, r2)
    P_ = P(n_, r1 ,r2)
    K_ = K(s1, P_)

    B_ = -(1/4) * rho(X,Y)**4 * B(n_, beta_, K_, P_)
    C_ = -X**2 * C(P_)
    D_ = -(1/2) * rho(X,Y)**2 * D(n_, P_)
    E_ = X * E(n_)
    F_ = X * rho(X,Y)**2 * F(n_, P_, K_, sigma_)

    total = B_ + C_ + D_ + E_ + F_

    abberation_2 = [B_, C_, D_, E_, F_, total]
    abberations_list_2.append(abberation_2)

np.shape(abberations_list_2)
```

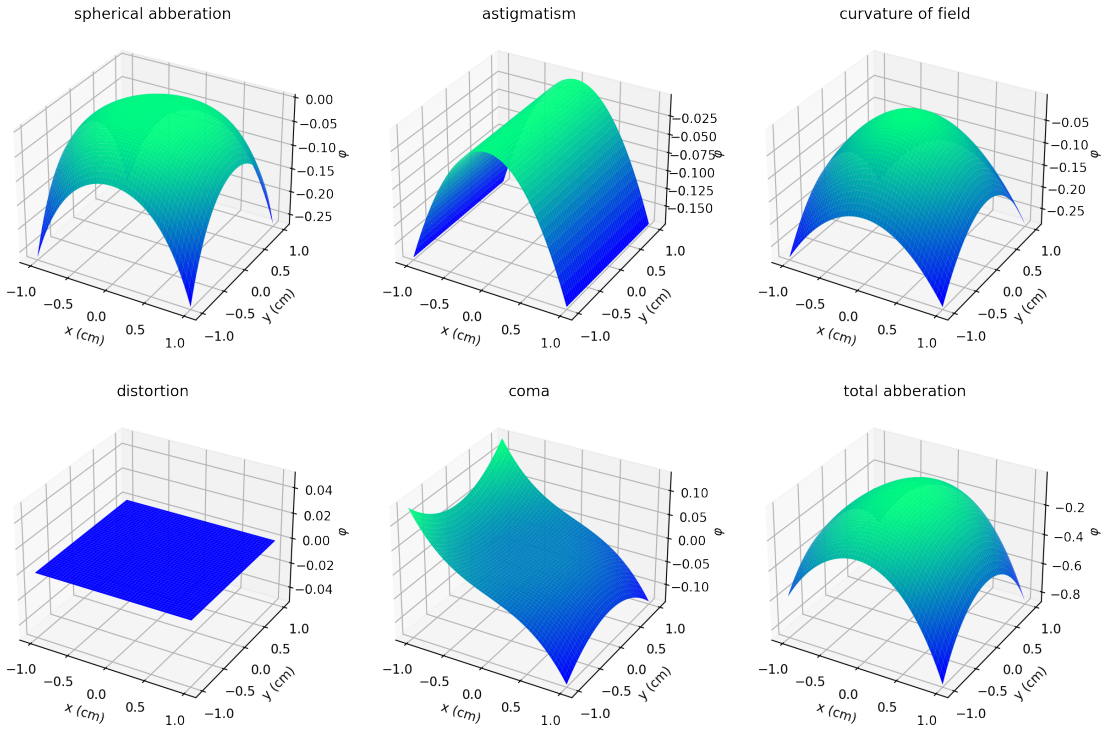
```
100%|                          | 3/3 [00:00<00:00,  9.00it/s]
```

[142]:
```
(3, 6, 1000, 1000)
```

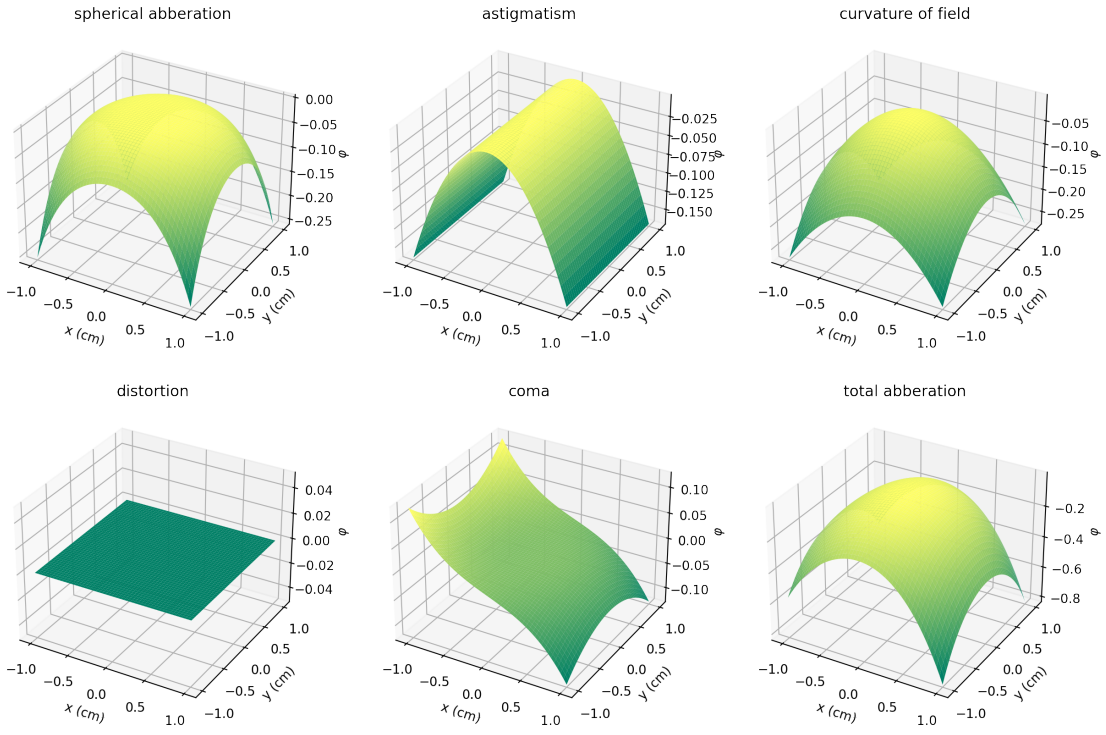
[143]:
```
for i in tqdm(range(n_lambda)):

    fig, axes = plt.subplots(figsize =(15,10), dpi = 200,
                      ncols=3, nrows = 2,
                      subplot_kw = dict ( projection = '3d'))

    for j, ax in enumerate(axes.ravel()):
        ax.plot_surface(X,Y, abberations_list_2[i][j], cmap = cmaps[i])
        ax.set_title(abberation_names[j])
        ax.set_xlabel('x (cm)')
        ax.set_ylabel('y (cm)')
        ax.set_zlabel('$\\varphi$')
    plt.suptitle('BaK1: $\lambda$ = %.1f nm' % lambda_list[i], fontsize = 20)
    plt.show()
```

```
  0%|                                                    | 0/3 [00:00<?, ?it/s]
```

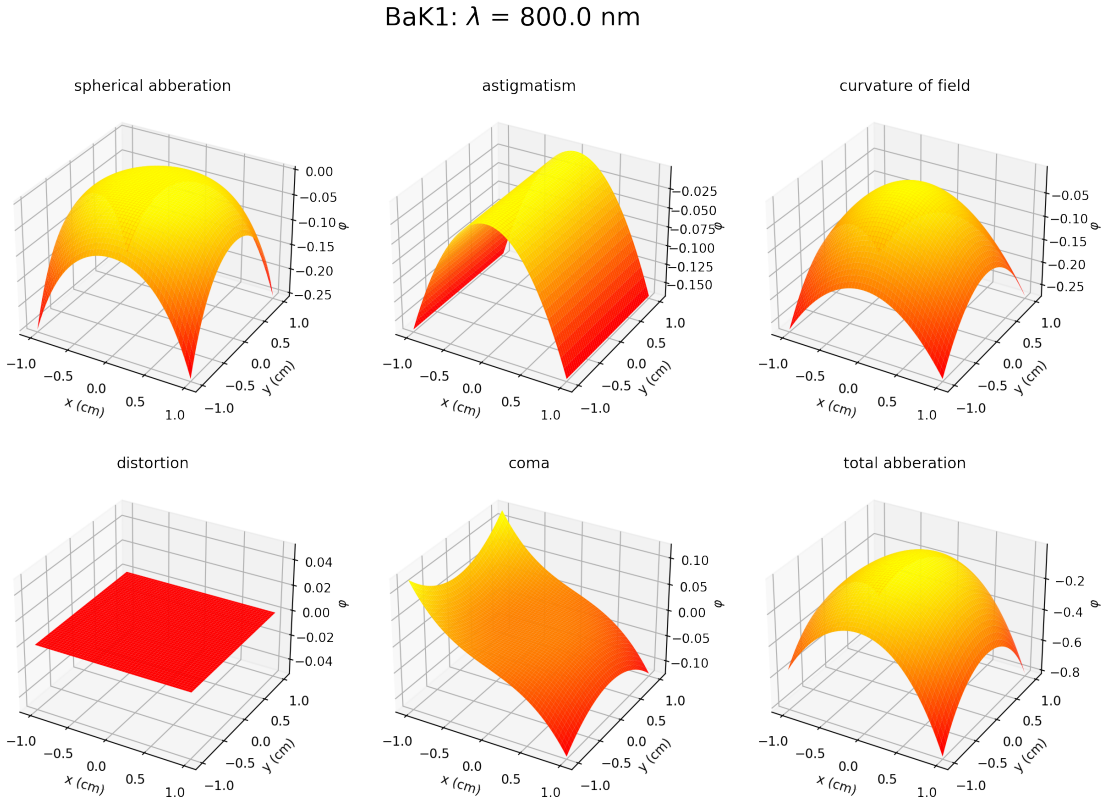
BaK1: λ = 400.0 nm



33%| | 1/3 [00:01<00:03, 1.87s/it]



67%| | 2/3 [00:03<00:01, 1.83s/it]



100%| | 3/3 [00:05<00:00, 1.79s/it]

---

4 Problem 7.3

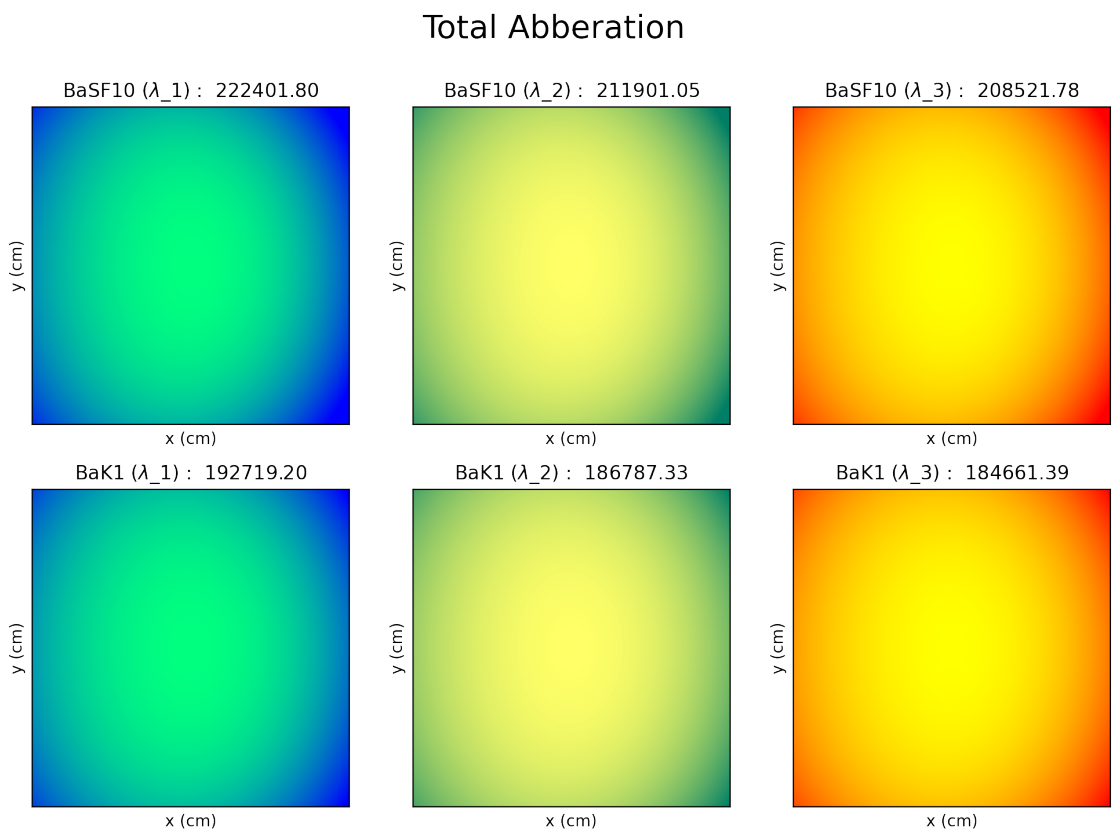
Which of the two lens materials produces less (total) aberration effect? Explain your answer.

[144]:
```
fig, axes = plt.subplots(figsize =(12,8), dpi = 200,
                         ncols=3, nrows = 2)

for i in tqdm(range(n_lambda)):
    total_1 = abberations_list[i][-1]
    sum_1 = np.abs(np.sum(total_1))
    axes[0,i].imshow(total_1, cmap = cmaps[i], vmin = -0.8, vmax = 0)
    axes[0,i].set_title('BaSF10 ($\lambda$_%.d) : % .2f' % (i+1, sum_1))

    total_2 = abberations_list_2[i][-1]
    sum_2 = np.abs(np.sum(total_2))
    axes[1,i].imshow(total_2, cmap = cmaps[i], vmin = -0.8, vmax = 0)
    axes[1,i].set_title('BaK1 ($\lambda$_%.d) : % .2f' % (i+1, sum_2))
```

```
for ax in axes.ravel():
    ax.set_xticks([])
    ax.set_yticks([])
    ax.set_xlabel('x (cm)')
    ax.set_ylabel('y (cm)')

plt.suptitle('Total Abberation', fontsize = 20)
```

```
100%|                          | 3/3 [00:00<00:00, 51.74it/s]
```

[144]: `Text(0.5, 0.98, 'Total Abberation')`



4.0.1 The absolute total abberation were summed up are shown, and BaK1 consistently yielded lesser total abberation effects across the three observation wavelengths ($\lambda = \{400\text{ nm}, 600\text{ nm}, 800\text{ nm}\}$).

---

5 Problem 7.4.

At what object distance s1 (from the apex of the first lens surface) will the thin lens produce no coma for the corresponding image produced? Plot the object distance value versus wavelength A in the range: 400 nm < A < 800 nm (oA = 0.1 nm) for each of your two dielectric materials. Briefly discuss your results.

*Note: For materials with operating ranges that are outside the visible spectrum (400 – 750 nm), please use the center wavelength of the stated range.

From `Section 5.6 Born & Wolf: Eq. 17`, coma is absent in the abberation whenever V = 0, which gives an equation for object distance s_1 as follows:

$$\frac{1}{s_1} = -\frac{1}{2}P + \frac{n+1}{2(n-1)(2n+1)}\sigma \tag{21}$$

where

$$P = (n-1)\left(\frac{1}{r_1} - \frac{1}{r_2}\right) \tag{22}$$

$$\sigma = (n-1)\left(\frac{1}{r_1} + \frac{1}{r_2}\right). \tag{23}$$

Simplifying the expression to obtain s_1,

$$\frac{1}{s_1} = -\frac{(n-1)}{2}\left(\frac{1}{r_1} - \frac{1}{r_2}\right) + \frac{n+1}{2(2n+1)}\left(\frac{1}{r_1} + \frac{1}{r_2}\right) \tag{24}$$

$$\frac{1}{s_1} = -\frac{(n-1)(r_2 - r_1)}{2r_1r_2} + \frac{(n+1)(r_1 + r_2)}{2r_1r_2(2n+1)} \tag{25}$$

$$\frac{1}{s_1} = \frac{-(n-1)(2n+1)(r_2 - r_1) + (n+1)(r_1 + r_2)}{2r_1r_2(2n+1)} \tag{26}$$

$$s_1 = \frac{2r_1r_2(2n+1)}{nr_1 + r_1 + nr_2 + r_2 - 2n^2r_2 + nr_2 + r_2 + 2n^2r_1 - nr_1 - r_1} \tag{27}$$

$$s_1 = \frac{2r_1r_2(2n+1)}{2n^2(r_1 - r_2) + 2r_2(n+1)} \tag{28}$$

$$s_1 = \frac{r_1r_2(2n+1)}{n^2(r_1 - r_2) + r_2(n+1)} \tag{29}$$

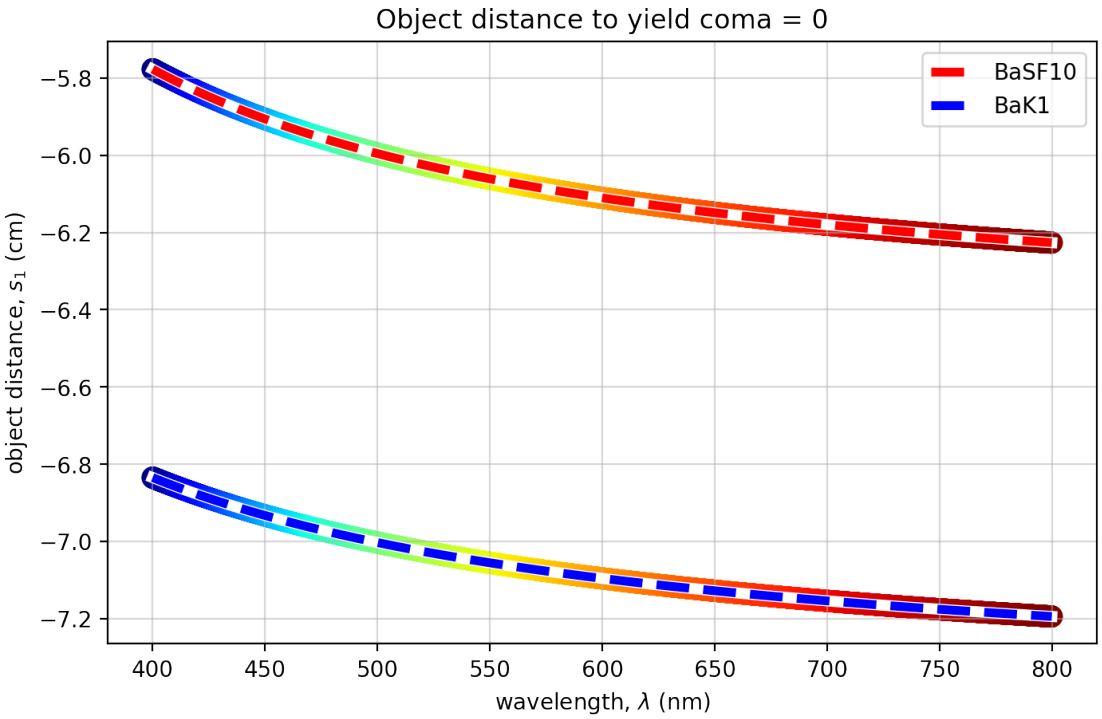
[145]:
```
d_lambda = 0.1
lambda_list = np.arange(400, 800, d_lambda)/1e3
```

[146]:
```
def s1(n, r1 , r2):
    num = r1 * r2 * (2*n + 1)
    denom = n**2 * (r1 - r2) + r2 * (n+1)
    return num/denom
```

[147]:
```
s1_BaSF10 = s1(BaSF10_n(lambda_list), r1, r2)
s1_BaK1 = s1(BaK1_n(lambda_list), r1, r2)
```

[148]:
```
fig, ax = plt.subplots(nrows=1,ncols=1, sharex='col',figsize=(8,5), dpi = 200)
fig.patch.set_facecolor('None')
plt.grid(alpha = 0.5)
plt.plot(lambda_list*1e3, s1_BaSF10, 'w-', lw = 5)
plt.plot(lambda_list*1e3, s1_BaSF10, 'r--', label = 'BaSF10', lw = 4)

plt.plot(lambda_list*1e3, s1_BaK1, 'w-', lw = 5)
plt.plot(lambda_list*1e3, s1_BaK1, 'b--', label = 'BaK1', lw = 4)

x,y = lambda_list*1e3, s1_BaSF10
colors = (y-min(y))/(max(y)-min(y))

plt.scatter(x, s1_BaSF10, c=cm.jet_r(colors), edgecolor='none', s=100)
plt.scatter(x, s1_BaK1, c=cm.jet_r(colors), edgecolor='none', s=100)



plt.xlabel('wavelength, $\lambda$ (nm)')
plt.ylabel('object distance, $s_1$ (cm)')
plt.legend()
plt.title('Object distance to yield coma = 0')
```

[148]:
```
Text(0.5, 1.0, 'Object distance to yield coma = 0')
```



The object distance solved were negative, indicating that object is virtual. Comparing BaSF10 and BaK1, both have increasing magnitude of object distance through longer wavelengths, but noticeably, BaK1 had larger object distances such that coma abberation is absent.