Example 1.

Sensitivity of hydraulic head at a point to spatially uniform hydraulic conductivity under steady state flow conditions

0. Forward model

Governing equation:

$$\begin{align} K\,b\,\frac{d^2h}{dx^2} + R = 0 \tag{1}\\ \tag{2}\end{align}$$

Boundary conditions:

$$\begin{align} -K\,b\,\frac{dh(x)}{dx} = 0\,, &\qquad x = 0 = \Gamma_2 \tag{3}\\ h(x) = h_{\Gamma_1}\,, &\qquad x = L = \Gamma_1 \tag{4}\\ \tag{5}\end{align}$$

Closed-form solution:

$$\begin{align} h(x) = h_L + \frac{R(L^2 - x^2)}{2\,K\,b} \tag{6}\\ \tag{7}\end{align}$$

Spatial derivatives of hydraulic head obtained from differentiation:

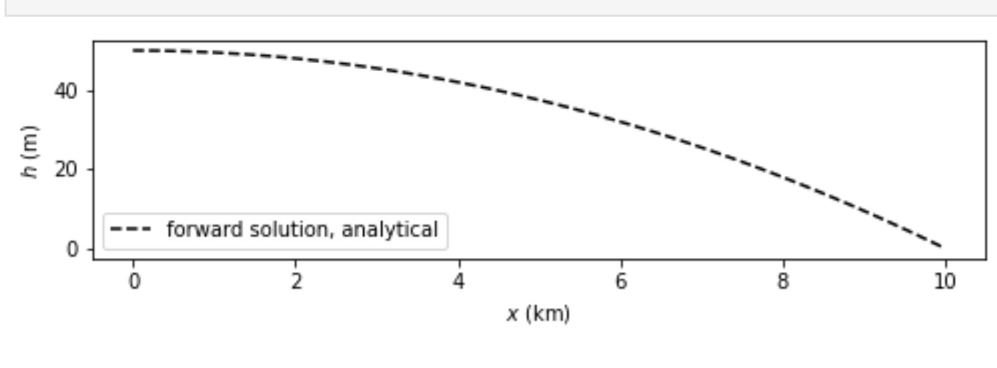
$$\frac{dh}{dx} = -\frac{R\,x}{K\,b}, \qquad \frac{d^2h}{dx^2} = -\frac{R}{K\,b} \tag{8}$$

In [72]:

```
from IPython.display import HTML, display
def set_background(color):
    script = (
        "var cell = this.closest('.code_cell');"
        "var editor = cell.querySelector('.input_area');"
        "editor.style.background='{}';"
        "this.parentNode.removeChild(this)").format(color)
    display(HTML('<img src onerror="{}">'.format(script)))
```

In [94]:

```
from warnings import filterwarnings
filterwarnings("ignore", category=DeprecationWarning)

import numpy as np

def h(x, K, R, b, L, BC1h):
    return R/K/2./b*(L**2.-x**2.)

K, R, b, L, BC1h, ocol = 10., 1e-1/1000., 10., 10000., 0., 5000
X = np.arange(L)
H0 = np.array([h(x, K, R, b, L, BC1h) for x in X])

%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
plt.subplots(figsize=[8,2])
plt.plot(X/1000., H0, 'k--', mfc='none', label='forward solution, analytical')
plt.xlabel('$x$ (km)')
plt.ylabel('$h$ (m)')
plt.legend(loc=3);
```

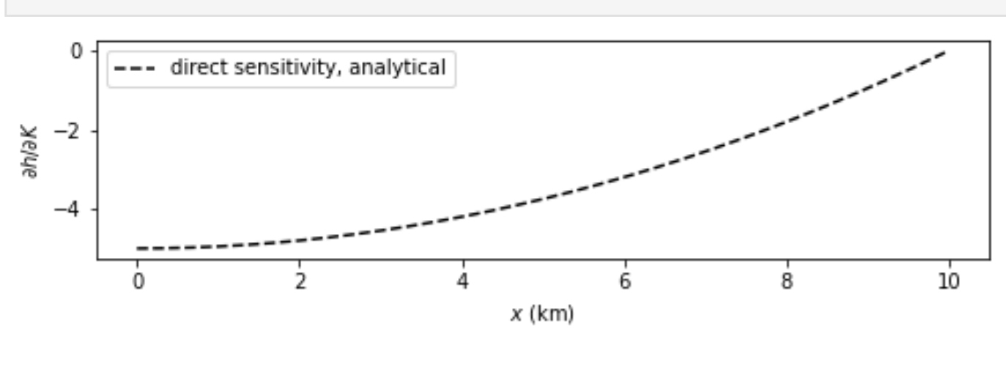


1. Direct sensitivity

$$\begin{align} \tag{9}\\ \frac{\partial h(x')}{\partial K} = \frac{R\,(x^2 - L^2)}{2\,K^2\,b} \tag{10}\\ \tag{11}\end{align}$$

In [74]:

```
dhdK = [R/K**2./2./b*(x**2.-L**2.) for x in X]
benchmark = dhdK[ocol]

%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
plt.subplots(figsize=[8,2])
plt.plot(X/1000., dhdK, 'k--', mfc='none', label='direct sensitivity, analytical')
plt.xlabel('$x$ (km)')
plt.ylabel(r'$\partial h / \partial K$')
plt.legend();
```



In [75]:

```
set_background('rgba(0, 200, 0, 0.2)')
print('%.6f'% benchmark)
```

```
-3.750000
```

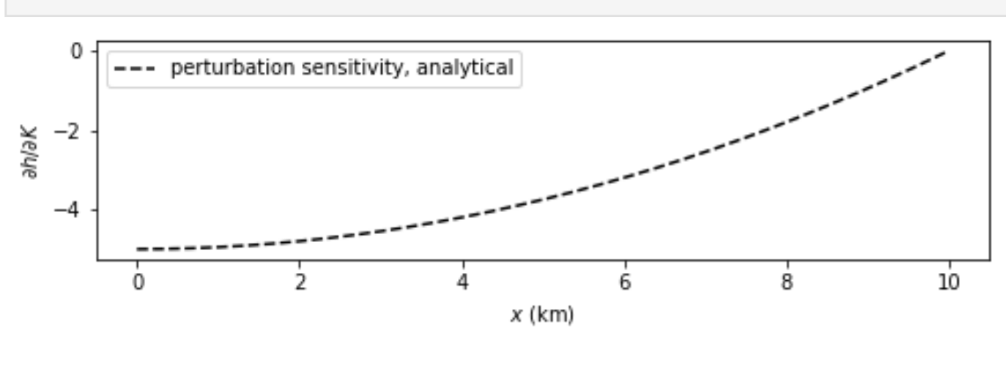
2. Perturbation sensitivity

$$\begin{align} \tag{12}\\ \frac{\partial h(x')}{\partial K} \approx \frac{h(x, K + \Delta K) - h(x, K)}{\Delta K} \tag{13}\\ \tag{14}\end{align}$$

2a. Analytical

In [76]:

```
dpar = 1e-4
H0 = np.array([h(x, K, R, b, L, BC1h) for x in X])
H1 = np.array([h(x, K+K*dpar, R, b, L, BC1h) for x in X])
dhdK = (H1-H0)/(K*dpar)

%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
plt.subplots(figsize=[8,2])
plt.plot(X/1000., dhdK, 'k--', mfc='none', label='perturbation sensitivity, analytical')
plt.xlabel('$x$ (km)')
plt.ylabel(r'$\partial h / \partial K$')
plt.legend();
```



In [77]:

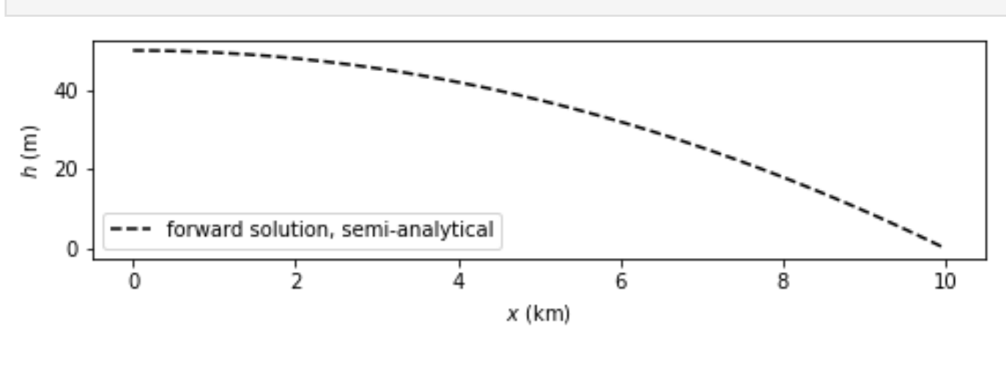
```
set_background('rgba(0, 200, 0, 0.2)')
print('%.6f (%.6f)'% (dhdK[ocol], benchmark))
```

```
-3.749625 (-3.750000)
```

2b. Semi-analytical

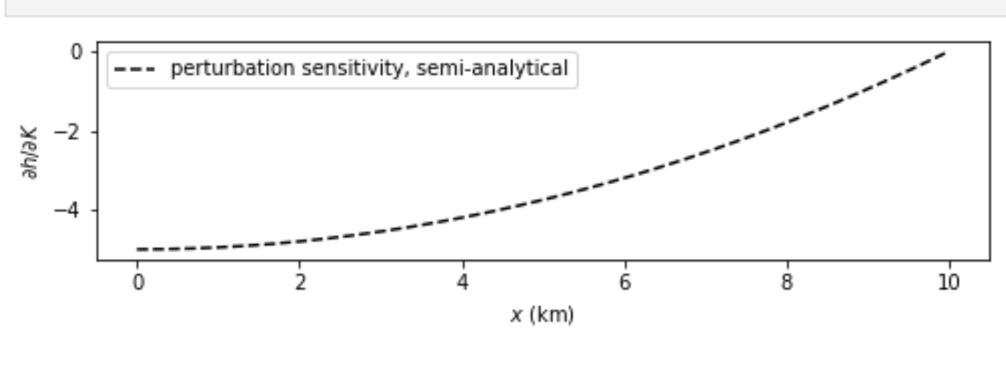
In [93]:

```
from os import getcwd, chdir
cwd = getcwd()
chdir(r'../../timml')
import timml
chdir(cwd)

M0 = timml.Model3D(kaq=K, z=[0., -b])
timml.ImpLineDoublet1D(M0, xld=0.)
timml.HeadLineSink1D(M0, xls=L, hls=0.)
timml.StripAreaSink(M0, 0., L, R)
M0.solve(silent=True)
H0 = M0.headalongline(X, 0.).flatten()

%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
plt.subplots(figsize=[8,2])
plt.plot(X/1000., H0, 'k--', mfc='none', label='forward solution, semi-analytical')
plt.xlabel('$x$ (km)')
plt.ylabel('$h$ (m)')
plt.legend(loc=3);
```



In [79]:

```
M1 = timml.Model3D(kaq=K+K*dpar, z=[0., -b])
timml.ImpLineDoublet1D(M1, xld=0.)
timml.HeadLineSink1D(M1, xls=L, hls=0.)
timml.StripAreaSink(M1, 0., L, R)
M1.solve(silent=True)
H1 = M1.headalongline(X, 0.).flatten()
dhdK = (H1-H0)/(K*dpar)

%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
plt.subplots(figsize=[8,2])
plt.plot(X/1000., dhdK, 'k--', mfc='none', label='perturbation sensitivity, semi-analytical')
plt.xlabel('$x$ (km)')
plt.ylabel(r'$\partial h / \partial K$')
plt.legend();
```



In [80]:

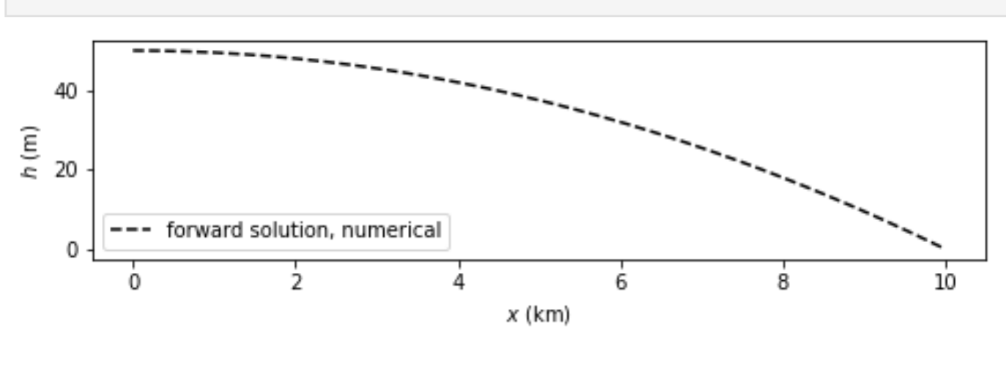
```
set_background('rgba(0, 200, 0, 0.2)')
print('%.6f (%.6f)'% (dhdK[ocol], benchmark))
```

```
-3.749625 (-3.750000)
```

2c. Numerical

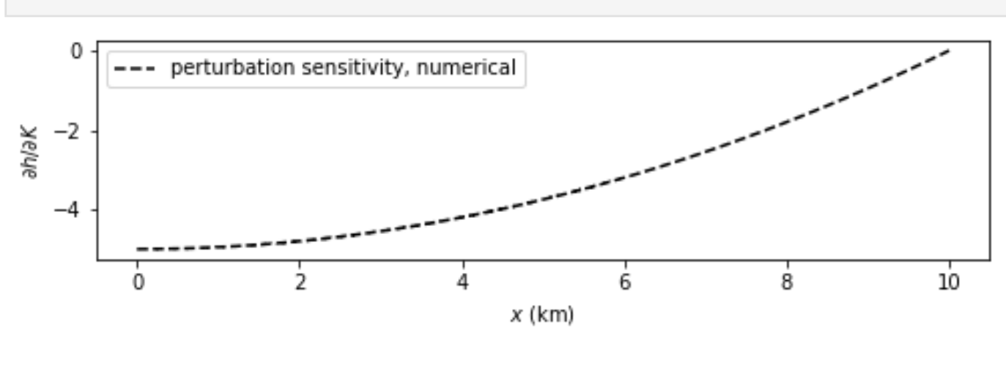
In [95]:

```
import flopy

nrow, ncol = 1, int(L)
M0 = flopy.modflow.Modflow(modelname='model', exe_name='../mf2005.exe')
flopy.modflow.ModflowDis(M0, nlay=1, nrow=nrow, ncol=ncol, nper=1, delr=1., delc=1., top=0., botm=-b, steady=Tr
                         perlen=1., nstp=1)
flopy.modflow.ModflowBas(M0, ibound=np.hstack([np.ones([nrow, ncol-1], dtype=int), -1*np.ones([1,1])]),
                         strt=BC1h*np.ones([nrow, ncol], dtype=float))
flopy.modflow.ModflowLpf(M0, hk=K, vka=-999., ss=-999., sy=-999., ipakcb=53)
flopy.modflow.ModflowRch(M0, nrchop=1, rech=R, ipakcb=53)
flopy.modflow.ModflowPcg(M0, hclose=1e-6, rclose=1e-6)
flopy.modflow.ModflowOc(M0, stress_period_data={(0,0): ['save head', 'save budget']})
M0.write_input()
success, buff = M0.run_model(silent=True)
H0 = flopy.utils.binaryfile.HeadFile('model.hds').get_data()[0,0,:]

%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
plt.subplots(figsize=[8,2])
plt.plot(X/1000., H0, 'k--', mfc='none', label='forward solution, numerical')
plt.xlabel('$x$ (km)')
plt.ylabel('$h$ (m)')
plt.legend(loc=3);
```



In [82]:

```
M1 = M0
flopy.modflow.ModflowLpf(M1, hk=K+K*dpar, vka=-999., ss=-999., sy=-999., ipakcb=53)
M1.write_input()
success, buff = M1.run_model(silent=True)
H1 = flopy.utils.binaryfile.HeadFile('model.hds').get_data()[0,0,:]
dhdK = (H1-H0)/(K*dpar)

%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
plt.subplots(figsize=[8,2])
plt.plot(X/1000., dhdK, 'k--', mfc='none', label='perturbation sensitivity, numerical')
plt.xlabel('$x$ (km)')
plt.ylabel(r'$\partial h / \partial K$')
plt.legend();
```



In [83]:

```
set_background('rgba(0, 200, 0, 0.2)')
print('%.6f (%.6f)'% (dhdK[ocol], benchmark))
```

```
-3.753662 (-3.750000)
```

3. Adjoint sensitivity

$$\frac{\partial h(x')}{\partial K} = \int_X \psi_1^*(x)\,b\,\frac{d^2h(x)}{dx^2}\,dx \tag{15}$$

Governing equation:

$$\begin{align} K\,b\,\frac{d\psi_1^*}{dx} + \frac{1}{2\,K\,b}\delta(x - x') = 0 \tag{16}\\ \tag{17}\end{align}$$

Boundary conditions:

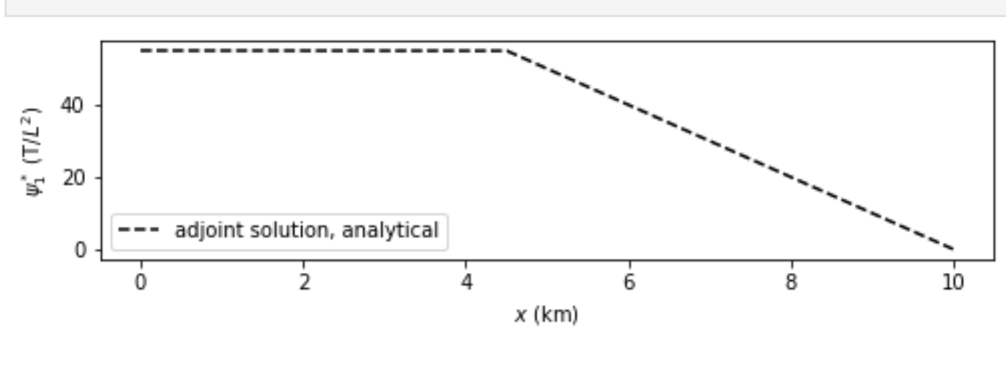
$$\begin{align} -K\,b\,\frac{d\psi_1^*(x)}{dx} = 0\,, &\qquad x = 0 = \Gamma_2 \tag{18}\\ \psi_1^*(x) = 0\,, &\qquad x = L = \Gamma_1 \tag{19}\\ \tag{20}\end{align}$$

Closed-form solution:

$$\psi_1^*(x) = \frac{1}{2\,K\,b}\left[H\left(x' - x\right)\left(L - x'\right) + H\left(x - x'\right)\left(L - x\right)\right] \tag{21}$$

In [96]:

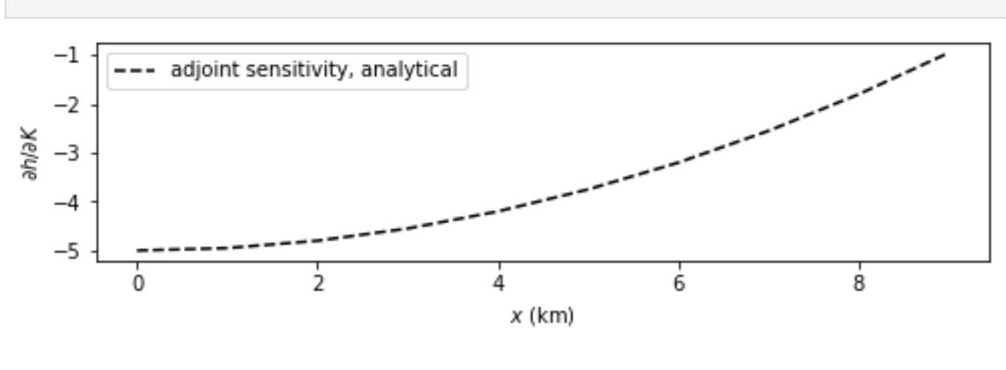
```
def a(x, xp, K, b, L):
    if x>xp:
        a = L-x
    else:
        a = L-xp
    return a/K/b
A = np.array([a(x, 4500., K, b, L) for x in X])

%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
plt.subplots(figsize=[8,2])
plt.plot(X/1000., A, 'k--', mfc='none', label='adjoint solution, analytical')
plt.xlabel('$x$ (km)')
plt.ylabel(r'$\psi^*_1$ (T/$L^2$)')
plt.legend(loc=3);
```



3a. Analytical

In [85]:

```
dhdK = [np.sum(np.array([a(x, xp, K, b, L) for x in X])*b*-R/(K*b)) for xp in X[::1000]]

%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
plt.subplots(figsize=[8,2])
plt.plot(X[::1000]/1000., dhdK, 'k--', mfc='none', label='adjoint sensitivity, analytical')
plt.xlabel('$x$ (km)')
plt.ylabel(r'$\partial h / \partial K$')
plt.legend();
```



In [86]:

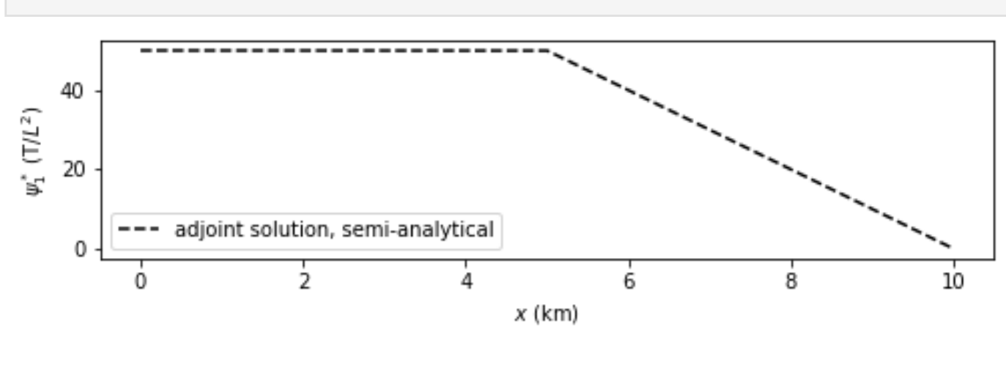
```
set_background('rgba(0, 200, 0, 0.2)')
print('%.6f (%.6f)'% (dhdK[int(np.where(X[::1000]==float(ocol))[0])], benchmark))
```

```
-3.750250 (-3.750000)
```

3b. Semi-analytical

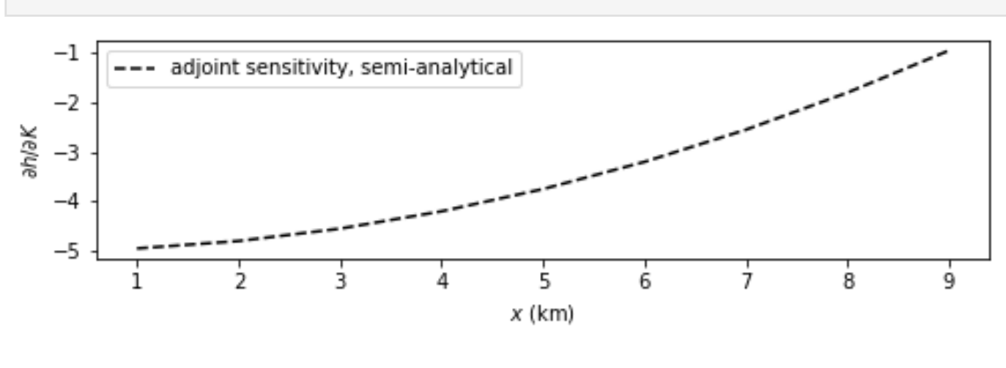
In [87]:

```
M0 = timml.Model3D(kaq=K, z=[0., -b])
timml.ImpLineDoublet1D(M0, xld=0.)
timml.HeadLineSink1D(M0, xls=L, hls=0.)
timml.LineSink1D(M0, xls=float(ocol), sigls=-1.)
M0.solve(silent=True)
A = M0.headalongline(X, 0.).flatten()

%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
plt.subplots(figsize=[8,2])
plt.plot(X/1000., A, 'k--', mfc='none', label='adjoint solution, semi-analytical')
plt.xlabel('$x$ (km)')
plt.ylabel(r'$\psi^*_1$ (T/$L^2$)')
plt.legend(loc=3);
```



In [97]:

```
dhdK = np.empty(0)
for xp in X[::1000]:
    M0 = timml.Model3D(kaq=K, z=[0., -b])
    timml.ImpLineDoublet1D(M0, xld=0.)
    timml.HeadLineSink1D(M0, xls=L, hls=0.)
    timml.LineSink1D(M0, xls=xp, sigls=-1.)
    M0.solve(silent=True)
    A = M0.headalongline(X, 0.).flatten()
    dhdK = np.append(dhdK, np.sum(A*b*-R/(K*b)))

%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
plt.subplots(figsize=[8,2])
plt.plot(X[::1000][1:]/1000., dhdK[1:], 'k--', mfc='none', label='adjoint sensitivity, semi-analytical')
plt.xlabel('$x$ (km)')
plt.ylabel(r'$\partial h / \partial K$')
plt.legend(loc=2);
```



In [89]:

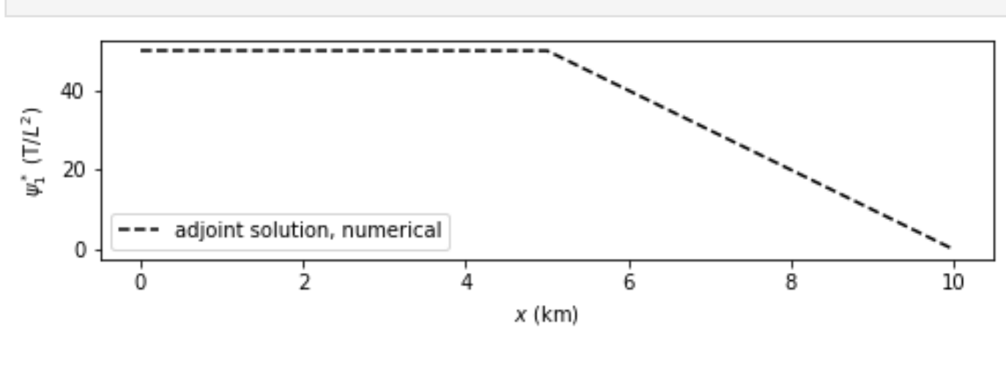
```
set_background('rgba(0, 200, 0, 0.2)')
print('%.6f (%.6f)'% (dhdK[int(np.where(X[::1000]==float(ocol))[0])], benchmark))
```

```
-3.750250 (-3.750000)
```

3c. Numerical, continuous

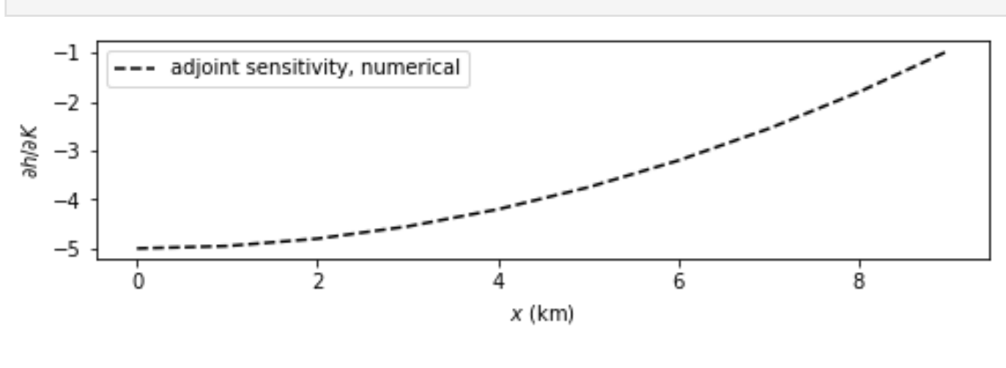
In [98]:

```
M0 = flopy.modflow.Modflow(modelname='model', exe_name='../mf2005.exe')
flopy.modflow.ModflowDis(M0, nlay=1, nrow=1, ncol=ncol, nper=1, delr=1., delc=1., top=0., botm=-b, steady=True,
                         perlen=1., nstp=1)
flopy.modflow.ModflowBas(M0, ibound=np.hstack([np.ones([nrow, ncol-1], dtype=int), -1*np.ones([1,1])]),
                         strt=np.zeros([nrow, ncol], dtype=float))
flopy.modflow.ModflowLpf(M0, hk=K, vka=-999., ss=-999., sy=-999., ipakcb=53)
flopy.modflow.ModflowWel(M0, stress_period_data={0: [0, 0, ocol, 1.]})
flopy.modflow.ModflowPcg(M0, hclose=1e-6, rclose=1e-6)
flopy.modflow.ModflowOc(M0, stress_period_data={(0,0): ['save head', 'save budget']})
M0.write_input()
success, buff = M0.run_model(silent=True)
A = flopy.utils.binaryfile.HeadFile('model.hds').get_data()[0,0,:]

%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
plt.subplots(figsize=[8,2])
plt.plot(X/1000., A, 'k--', mfc='none', label='adjoint solution, numerical')
plt.xlabel('$x$ (km)')
plt.ylabel(r'$\psi^*_1$ (T/$L^2$)')
plt.legend(loc=3);
```



In [91]:

```
dhdK = np.empty(0)
for oc,xp in enumerate(X[::1000]):
    M1 = M0
    flopy.modflow.ModflowWel(M1, stress_period_data={0: [0, 0, oc*1000, 1.]})
    M1.write_input()
    success, buff = M1.run_model(silent=True)
    A = flopy.utils.binaryfile.HeadFile('model.hds').get_data()[0,0,:]
    dhdK = np.append(dhdK, np.sum(A*b*-R/(K*b)))

%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
f,s = plt.subplots(figsize=[8,2])
plt.plot(X[::1000]/1000., dhdK, 'k--', mfc='none', label='adjoint sensitivity, numerical')
plt.xlabel('$x$ (km)')
plt.ylabel(r'$\partial h / \partial K$')
plt.legend();
```



In [92]:

```
set_background('rgba(0, 200, 0, 0.2)')
print('%.6f (%.6f)'% (dhdK[int(np.where(X[::1000]==float(ocol))[0])], benchmark))
```

```
-3.749250 (-3.750000)
```

3d. Numerical, discrete

In [ ]:

```
dhdK = np.empty(0)
M1 = M0
flopy.modflow.ModflowWel(M1, stress_period_data={0: [0, 0, ocol, 1.]})
M1.write_input()
success, buff = M1.run_model(silent=True)
psi = np.reshape(flopy.utils.binaryfile.HeadFile('model.hds').get_data()[0,0,:], [1,ncol])
#psi = psi[0, range(500, np.shape(psi)[1], 1000)]
```

In [ ]:

```
import flopy
from os import system

nrow, ncol = 1, 10000
M0 = flopy.modflow.Modflow(modelname='model', exe_name='../mf2005.exe')
flopy.modflow.ModflowDis(M0, nlay=1, nrow=nrow, ncol=ncol, nper=1, delr=1., delc=1., top=0., botm=-b, steady=Tr
                         perlen=1., nstp=1)
flopy.modflow.ModflowBas(M0, ibound=np.hstack([np.ones([nrow, ncol-1], dtype=int), -1*np.ones([1,1])]),
                         strt=BC1h*np.ones([nrow, ncol], dtype=float))
flopy.modflow.ModflowLpf(M0, hk=K, vka=-999., ss=-999., sy=-999., ipakcb=53)
flopy.modflow.ModflowRch(M0, nrchop=1, rech=R, ipakcb=53)
flopy.modflow.ModflowPcg(M0, hclose=1e-6, rclose=1e-6)
flopy.modflow.ModflowOc(M0, stress_period_data={(0,0): ['save head', 'save budget']})
M0.write_input()
#success, buff = M0.run_model(silent=True)
system('mf2005_stream.exe < model.in')
h = np.reshape(flopy.utils.binaryfile.HeadFile('model.hds').get_data()[0,0,:], [1,ncol])

%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
plt.subplots(figsize=[8,2])
plt.plot(X/1000., h[0, range(500, np.shape(h)[1], 1000)], 'ko--', mfc='none', label='forward solution, numerica
plt.xlabel('$x$ (km)')
plt.ylabel('$h$ (m)')
plt.legend();
```

In [ ]:

```
print(h[0, range(500, np.shape(h)[1], 1000)])
h = h.T
```

Assemble A matrix and RHS vector:

In [ ]:

```
CR = np.reshape(np.loadtxt('CR.arr').flatten(), [1,ncol])
CR = np.vstack([np.zeros(ncol),       CR, np.zeros(ncol)])
CR = np.hstack([np.zeros([nrow+2,1]), CR, np.zeros([nrow+2,1])])
CC  = np.zeros(np.shape(CR))
A = np.zeros([nrow*ncol, nrow*ncol])
i = 0
for r in range(1, nrow+1):
    for c in range(1, ncol+1):
        if r==nrow and c==ncol:
            continue
        if i-ocol > -1:
            A[i, i-ocol] = CC[r-1,c]
        if i-1 > -1:
            A[i, i-1] = CR[r,c-1]
        A[i, i] = -CC[r-1,c]-CR[r,c-1]-CC[r,c]-CR[r,c]
        if i+1 < nrow*ncol-1:
            A[i, i+1] = CR[r,c]
        if i+ocol < nrow*ncol-1:
            A[i, i+ocol] = CC[r,c]
        i +=1
#A = A[:-1,:-1]
print(A)
```

In [ ]:

```
RHS = np.reshape(np.loadtxt('RHS.arr'), [1,ncol]).T
print(RHS)
```

In [ ]:

```
nrow, ncol, ocol = 1, 10000, 5000
M0 = flopy.modflow.Modflow(modelname='model', exe_name='../mf2005.exe')
flopy.modflow.ModflowDis(M0, nlay=1, nrow=nrow, ncol=ncol, nper=1, delr=1., delc=1., top=0., botm=-b, steady=Tr
                         perlen=1., nstp=1)
flopy.modflow.ModflowBas(M0, ibound=np.hstack([np.ones([nrow, ncol-1], dtype=int), -1*np.ones([1,1])]),
                         strt=BC1h*np.ones([nrow, ncol], dtype=float))
flopy.modflow.ModflowLpf(M0, hk=K, vka=-999., ss=-999., sy=-999., ipakcb=53)
flopy.modflow.ModflowRch(M0, nrchop=1, rech=R, ipakcb=53)
flopy.modflow.ModflowPcg(M0, hclose=1e-6, rclose=1e-6)
flopy.modflow.ModflowOc(M0, stress_period_data={(0,0): ['save head', 'save budget']})
M0.write_input()
#success, buff = M0.run_model(silent=True)
system('mf2005_stream.exe < model.in')

h = np.reshape(flopy.utils.binaryfile.HeadFile('model.hds').get_data()[0,0,:], [1,ncol]).T
CR = np.reshape(np.loadtxt('CR.arr').flatten(), [1,ncol])
CR = np.vstack([np.zeros(ncol),       CR, np.zeros(ncol)])
CR = np.hstack([np.zeros([nrow+2,1]), CR, np.zeros([nrow+2,1])])
CC  = np.zeros(np.shape(CR))
A = np.zeros([nrow*ncol, nrow*ncol])
i = 0
for r in range(1, nrow+1):
    for c in range(1, ncol+1):
        if i-1 > -1:
            A[i,i-1] = CR[r,c-1]
        A[i,i] = -CR[r,c-1]-CR[r,c]
        if i+1 < nrow*ncol-1:
            A[i,i+1] = CR[r,c]
        i +=1
A[-1,-1] = 0.
A0 = A
print(A0)
```

In [ ]:

```
dpar = 1e-4
nrow, ncol, ocol = 1, 10000, 5000
M0 = flopy.modflow.Modflow(modelname='model', exe_name='../mf2005.exe')
flopy.modflow.ModflowDis(M0, nlay=1, nrow=nrow, ncol=ncol, nper=1, delr=1., delc=1., top=0., botm=-b, steady=Tr
                         perlen=1., nstp=1)
flopy.modflow.ModflowBas(M0, ibound=np.hstack([np.ones([nrow, ncol-1], dtype=int), -1*np.ones([1,1])]),
                         strt=BC1h*np.ones([nrow, ncol], dtype=float))
flopy.modflow.ModflowLpf(M0, hk=K+K*dpar, vka=-999., ss=-999., sy=-999., ipakcb=53)
flopy.modflow.ModflowRch(M0, nrchop=1, rech=R, ipakcb=53)
flopy.modflow.ModflowPcg(M0, hclose=1e-6, rclose=1e-6)
flopy.modflow.ModflowOc(M0, stress_period_data={(0,0): ['save head', 'save budget']})
M0.write_input()
#success, buff = M0.run_model(silent=True)
system('mf2005_stream.exe < model.in')

h = np.reshape(flopy.utils.binaryfile.HeadFile('model.hds').get_data()[0,0,:], [1,ncol]).T
CR = np.reshape(np.loadtxt('CR.arr').flatten(), [1,ncol])
CR = np.vstack([np.zeros(ncol),       CR, np.zeros(ncol)])
CR = np.hstack([np.zeros([nrow+2,1]), CR, np.zeros([nrow+2,1])])
CC  = np.zeros(np.shape(CR))
A = np.zeros([nrow*ncol, nrow*ncol])
i = 0
for r in range(1, nrow+1):
    for c in range(1, ncol+1):
        if i-1 > -1:
            A[i,i-1] = CR[r,c-1]
        A[i,i] = -CR[r,c-1]-CR[r,c]
        if i+1 < nrow*ncol-1:
            A[i,i+1] = CR[r,c]
        i +=1
A1 = A
print(A1)
```

In [ ]:

```
dAdK = (A1-A0)/(K*dpar)
```

In [ ]:

```
dAdK
```

In [ ]:

```
psi
```

In [ ]:

```
dAdK
```

In [ ]:

```
h
```

In [ ]:

```
np.float(-psi@dAdK@h)
```

In [ ]:

```
np.shape(psi), np.shape(dAdK), np.shape(h)
```

In [ ]:

```

```