Example 2.

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

0. Forward model

Governing equation:

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

Boundary conditions:

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

Closed-form solution:

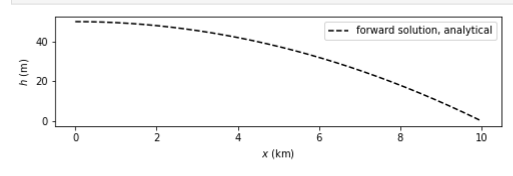
$$h(x) = h_L + \frac{R(L^2 - x^2)}{2\,K\,b} \tag{6}$$
$$\tag{7}$$

Spatial derivatives from differentiation:

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

In [22]:
```python
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)))
```

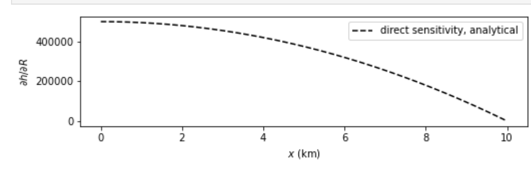
In [21]:
```python
from warnings import filterwarnings
filterwarnings("ignore", category=DeprecationWarning)

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();
```



1. Direct sensitivity

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

In [23]:
```python
dhdR = [(L**2.-x**2.)/(2.*K*b) for x in X]
benchmark = dhdR[ocol]

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



In [24]:
```python
print('%.6f (%.6f)'% (dhdR[ocol], benchmark))
```

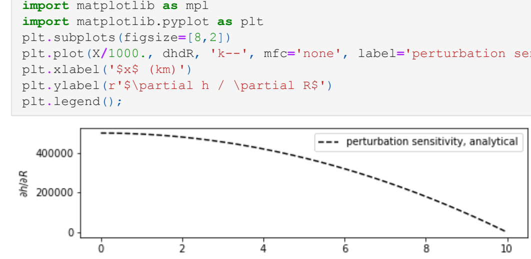
```
375000.000000 (375000.000000)
```

2. Perturbation sensitivity

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

2a. Analytical

In [25]:
```python
dpar = 1e-4
H0 = np.array([h(x, K, R, b, L, BC1h) for x in X])
H1 = np.array([h(x, K, R+R*dpar, b, L, BC1h) for x in X])
dhdR = (H1-H0)/(R*dpar)

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

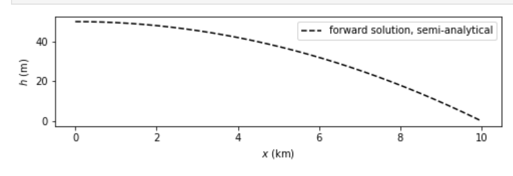


In [26]:
```python
print('%.6f (%.6f)'% (dhdR[ocol], benchmark))
```

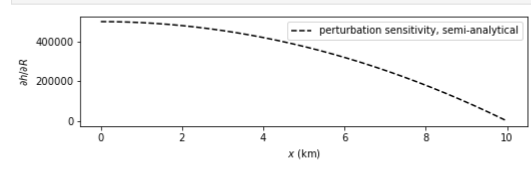
```
374999.999999 (375000.000000)
```

2b. Semi-analytical

In [27]:
```python
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();
```



In [28]:
```python
M1 = timml.Model3D(kaq=K, z=[0., -b])
timml.ImpLineDoublet1D(M1, xld=0.)
timml.HeadLineSink1D(M1, xls=L, hls=0.)
timml.StripAreaSink(M1, 0., L, R+R*dpar)
M1.solve(silent=True)
H1 = M1.headalongline(X, 0.).flatten()
dhdR = (H1-H0)/(R*dpar)

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

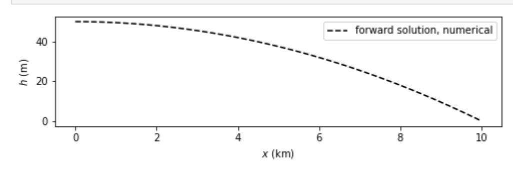


In [29]:
```python
print('%.6f (%.6f)'% (dhdR[ocol], benchmark))
```

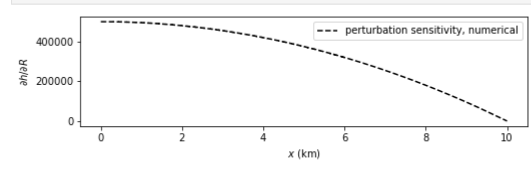
```
375000.000000 (375000.000000)
```

2c. Numerical

In [30]:
```python
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();
```



In [31]:
```python
M1 = M0
flopy.modflow.ModflowRch(M1, nrchop=1, rech=R+R*dpar, ipakcb=53)
M1.write_input()
success, buff = M1.run_model(silent=True)
H1 = flopy.utils.binaryfile.HeadFile('model.hds').get_data()[0,0,:]
dhdR = (H1-H0)/(R*dpar)

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



In [32]:
```python
print('%.6f (%.6f)'% (dhdR[ocol], benchmark))
```

```
374984.750000 (375000.000000)
```

3. Adjoint sensitivity

$$\frac{\partial h(x')}{\partial R} = \int_X \psi_1^*(x)\;dx \tag{15}$$

Governing equation:

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

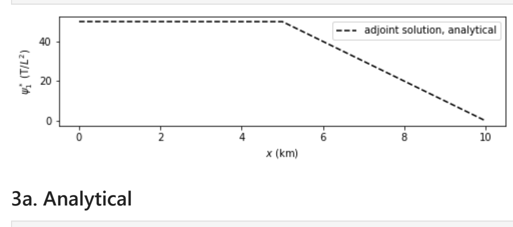
Boundary conditions:

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

Closed-form solution:

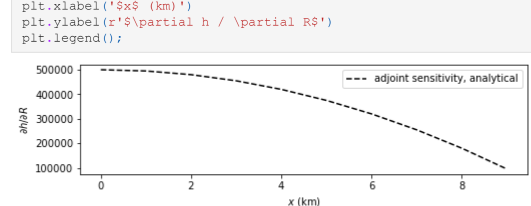
$$\psi_1^*(x) = \frac{1}{2\,K\,b}\left[H\left(x' - x\right)\left(L - x'\right) + H\left(x - x'\right)\left(L - x\right)\right] \tag{21}$$

In [33]:
```python
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, float(ocol), 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();
```



3a. Analytical

In [34]:
```python
dhdR = [np.sum([a(x, xp, K, b, L) for x in X]) 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., dhdR, 'k--', mfc='none', label='adjoint sensitivity, analytical')
plt.xlabel('$x$ (km)')
plt.ylabel(r'$\partial h / \partial R$')
plt.legend();
```

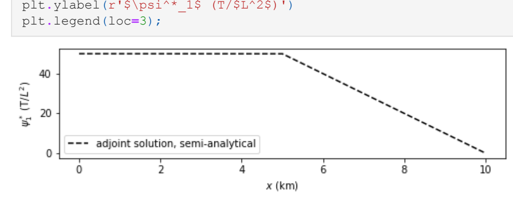


In [36]:
```python
print('%.6f (%.6f)'% (dhdR[int(np.where(X[::1000]==float(ocol))[0])], benchmark))
```

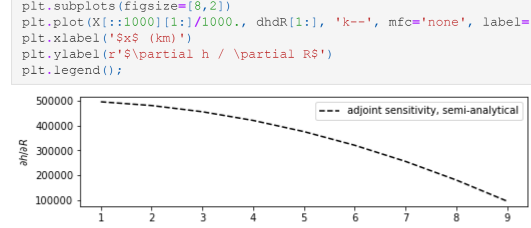
```
375025.000000 (375000.000000)
```

3b. Semi-analytical

In [37]:
```python
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 [40]:
```python
dhdR = 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()
    dhdR = np.append(dhdR, np.sum(A))

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

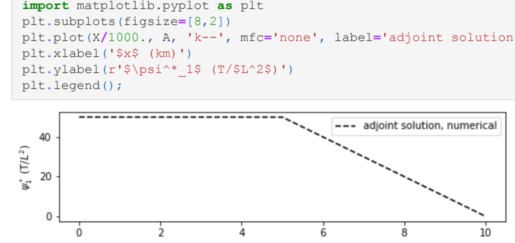


In [41]:
```python
print('%.6f (%.6f)'% (dhdR[int(np.where(X[::1000]==float(ocol))[0])], benchmark))
```

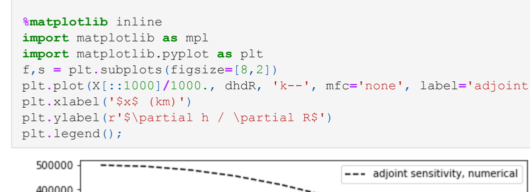
```
375025.000000 (375000.000000)
```

3c. Numerical

In [42]:
```python
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();
```



In [45]:
```python
dhdR = 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,:]
    dhdR = np.append(dhdR, np.sum(A))

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



In [46]:
```python
print('%.6f (%.6f)'% (dhdR[int(np.where(X[::1000]==float(ocol))[0])], benchmark))
```

```
374924.968750 (375000.000000)
```

In [ ]: