Example 8.

Sensitivity of hydraulic head at a point and at a discrete time to spatially uniform specific storage under transient flow conditions

0. Forward model

Governing equation:

$$K\,b\,\frac{\partial^2 h}{\partial x^2} + R = S_s\,b\,\frac{\partial h}{\partial t} \tag{1}$$
$$\tag{2}$$

Boundary conditions:

$$-K\,b\,\frac{dh(x)}{dx} = 0\,, \qquad x = 0 = \Gamma_2 \tag{3}$$
$$h(x,t) = h_{\Gamma_{1_L}}\,, \qquad x = L = \Gamma_{1_L} \tag{4}$$
$$\tag{5}$$

Initial conditions:

$$h(x,t) = h_0\,, \qquad t = 0 \tag{6}$$
$$\tag{7}$$

Closed-form solution:

$$\text{Not available} \tag{8}$$
$$\tag{9}$$

Spatial derivatives from differentiation:

$$\text{Not available} \tag{10}$$
$$\tag{11}$$

In [91]:
```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 [92]:
```python
from warnings import filterwarnings
filterwarnings("ignore", category=DeprecationWarning)

import numpy as np

K, Ss, R, b, L, BC1h, ocol, nper = 10., 1e-2, 1e-1/1000., 10., 10000., 0., 5000, 1000
X = np.arange(L)
```

1. Direct sensitivity

$$\text{Not available} \tag{12}$$

2. Perturbation sensitivity

$$\tag{13}$$
$$\frac{\partial h(x')}{\partial S_s} \approx \frac{h(x, S_s + \Delta S_s) - h(x, S_s)}{\Delta S_s} \tag{14}$$
$$\tag{15}$$

2a. Analytical

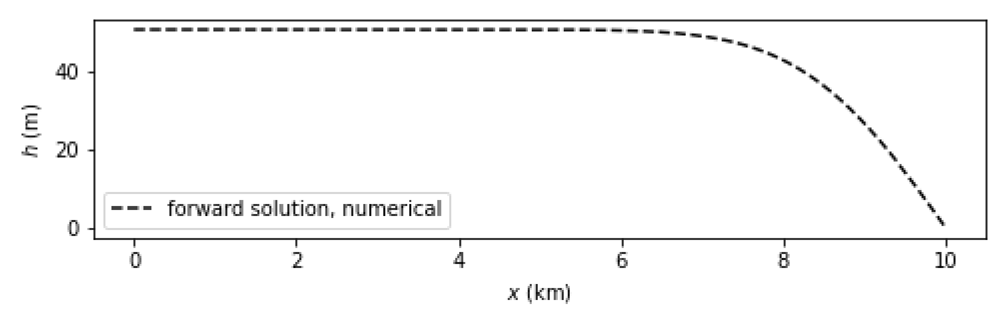
$$\text{Not available} \tag{16}$$

2b. Semi-analytical

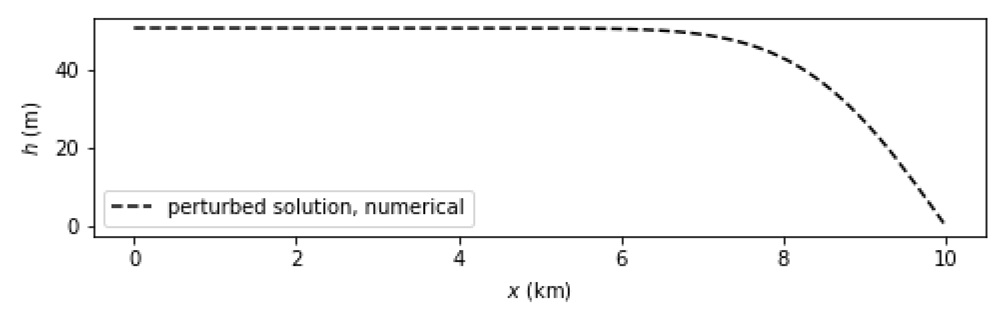
$$\text{Not available} \tag{17}$$

2c. Numerical

In [93]:
```python
import flopy

dpar = 1e-2
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=nper, delr=1., delc=1., top=0., botm=-b,
                         steady=False, perlen=np.ones(nper), nstp=1)
flopy.modflow.ModflowBas(M0, ibound=np.hstack([np.ones([nrow, ncol-1], dtype=int), -1*np.ones([1,1])]),
                         strt=np.hstack([50.*np.ones([nrow, ncol-1], dtype=float), np.atleast_2d([0.])]))
flopy.modflow.ModflowLpf(M0, hk=K, vka=-999., ss=Ss, sy=-999., ipakcb=53)
flopy.modflow.ModflowRch(M0, nrchop=1, rech={0:R}, ipakcb=53)
flopy.modflow.ModflowPcg(M0, hclose=1e-6, rclose=1e-6)
flopy.modflow.ModflowOc(M0, stress_period_data={(i,0): ['save head'] for i in range(nper)})
M0.write_input()
success, buff = M0.run_model(silent=True)
H0 = flopy.utils.binaryfile.HeadFile('model.hds').get_data(kstpkper=[0,nper-1])[0,0,:]
hdt = np.gradient(np.array([flopy.utils.binaryfile.HeadFile('model.hds').get_data(kstpkper=[0,i])[0,0,:]
                            for i in range(nper)]), axis=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 [94]:
```python
M1 = M0
flopy.modflow.ModflowLpf(M1, hk=K, vka=-999., ss=Ss+Ss*dpar, sy=-999., ipakcb=53)
M1.write_input()
success, buff = M1.run_model(silent=True)
H1 = flopy.utils.binaryfile.HeadFile('model.hds').get_data(kstpkper=[0,nper-1])[0,0,:]
dhdSs = (H1-H0)/(Ss*dpar)
benchmark = dhdSs[ocol]

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



In [96]:
```python
set_background('rgba(0, 200, 0, 0.2)')
print('%.6f'% dhdSs[ocol])
```

```
-86.555481
```

3. Adjoint sensitivity

$$\frac{\partial h(x')}{\partial S_s} = \int_T \int_X \psi_1^*(x, t' - t)\,\frac{dh(x,t)}{dt}\,dx\,dt \tag{18}$$

Governing equation:

$$K\,b\,\frac{\partial^2 \psi_1^*}{\partial x^2} + \delta(x - x')\,\delta(t - t') = -S_s\,b\,\frac{\partial \psi_1^*}{\partial \tau} \tag{19}$$
$$\tag{20}$$

Boundary conditions:

$$-K\,b\,\frac{d\psi_1^*}{dx} = 0\,, \qquad x = 0 = \Gamma_2 \tag{21}$$
$$\psi_1^*(x,t) = 0\,, \qquad x = L = \Gamma_{1_L} \tag{22}$$
$$\tag{23}$$

Terminal conditions:

$$\psi_1^*(x,t) = 0\,, \qquad t = t_{final} \tag{24}$$
$$\tag{25}$$

Closed-form solution:

$$\text{Not available} \tag{26}$$

3a. Analytical

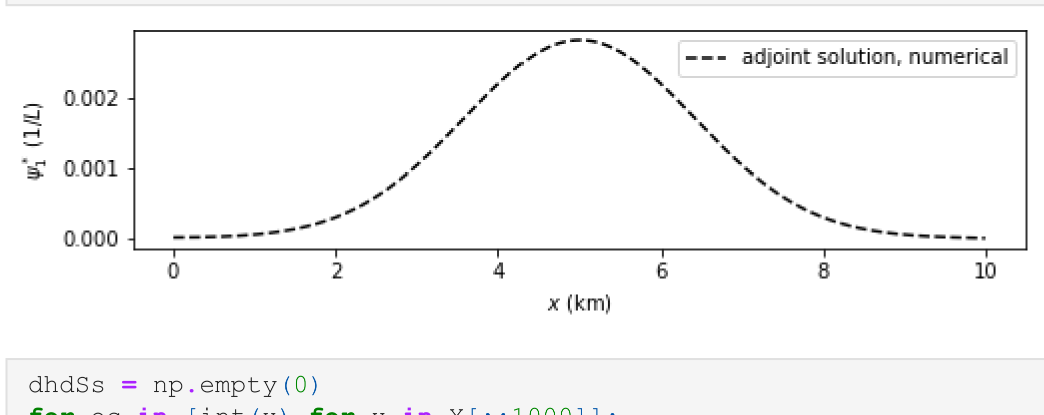
$$\text{Not available} \tag{27}$$

3b. Semi-analytical

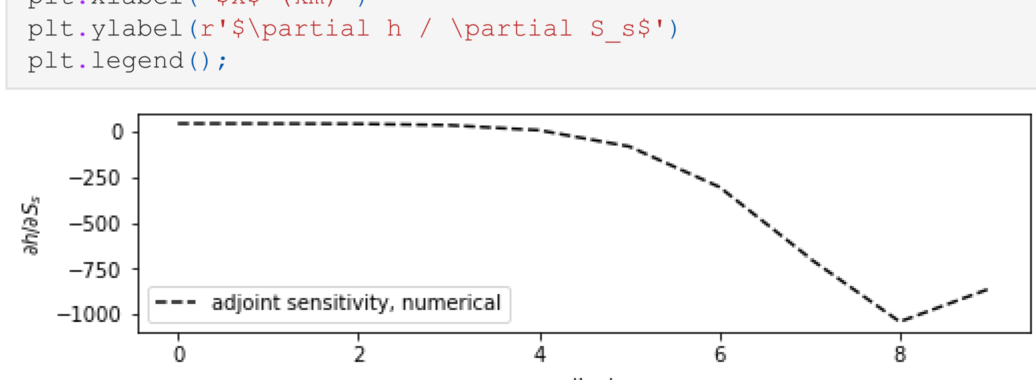
$$\text{Not available} \tag{28}$$

3c. Numerical

In [97]:
```python
M0 = flopy.modflow.Modflow(modelname='model', exe_name='../mf2005.exe')
flopy.modflow.ModflowDis(M0, nlay=1, nrow=1, ncol=ncol, nper=nper, delr=1., delc=1., top=0., botm=-b,
                         steady=False, perlen=np.ones(nper), nstp=1)
strt = np.zeros([nrow, ncol], dtype=float)
strt[0,ocol] = 1./Ss/b
flopy.modflow.ModflowBas(M0, ibound=np.hstack([np.ones([nrow, ncol-1], dtype=int), -1*np.ones([1,1])]),
                         strt=strt)
flopy.modflow.ModflowLpf(M0, hk=K, vka=-999., ss=Ss, sy=-999., ipakcb=53)
flopy.modflow.ModflowPcg(M0, hclose=1e-6, rclose=1e-6)
flopy.modflow.ModflowOc(M0, stress_period_data={(i,0): ['save head'] for i in range(nper)})
M0.write_input()
success, buff = M0.run_model(silent=True)
A = np.array([flopy.utils.binaryfile.HeadFile('model.hds').get_data(kstpkper=[0,i])[0,0,:]
              for i in range(nper)])[::-1,:]

%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
f,s = plt.subplots(figsize=[8,2])
plt.plot(X/1000., np.ravel(A[0,:]), 'k--', mfc='none', label='adjoint solution, numerical')
plt.xlabel('$x$ (km)')
plt.ylabel(r'$\psi^*_1$ (1/$L$)')
plt.legend();
```



In [100…
```python
dhdSs = np.empty(0)
for oc in [int(x) for x in X[::1000]]:
    M1 = M0
    strt = np.zeros([nrow, ncol], dtype=float)
    strt[0,oc] = 1./Ss/b
    flopy.modflow.ModflowBas(M1, ibound=np.hstack([np.ones([nrow, ncol-1], dtype=int), -1*np.ones([1,1])]),
                             strt=strt)
    M1.write_input()
    success, buff = M1.run_model(silent=True)
    A = flopy.utils.binaryfile.HeadFile('model.hds').get_data(kstpkper=[0,nper-1])[0,0,:]
    dhdSs = np.append(dhdSs, np.sum(np.sum(np.multiply(A, dhdt), axis=0), axis=0)*b)

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



In [101…
```python
set_background('rgba(0, 200, 0, 0.2)')
print('%.6f (%.6f)'% (dhdSs[int(np.where(X[::1000]==float(ocol))[0])], benchmark))
```

```
-85.246525 (-86.555481)
```

In [ ]: