Bruggeman solution 223_02

December 19, 2024

1 Intro, aanleiding

Ik zocht het verloop van de onttrekking rond een cikelvormige bouwput in een oneindig uitgestrekt pakket bij vaste verlaging op zijn rand $r = R$. Ik heb dit probleem gereduceerd tot een onttrekking op een cilinder met radius R als gevolg van een plotselinge verandering met gegeven h op $t = 0$. De oplossing voor dit probleem wordt gegeven door Bruggeman (1999) als oplosing 223_02, p163:

$$\phi(r,t) = h\left\{1 - \frac{2}{\pi}\int_{u=0}^{\infty}\frac{1}{u}h(u,r)\exp\left(-\frac{u^2t}{\beta^2R^2}\right)du\right\}$$

met

$$h(u,r/R) = \frac{J_0(u)Y_0(\frac{r}{R}u) - Y_0(u)J_0(\frac{r}{R}u)}{J_0^2(u) + Y_0^2(u)}$$

en Y_0 en J_0 de bekende oscilerende Besselfuncties.

Het debiet $Q\left(t,r\right)$ wordt op de standaard wijze verkregen als

$$Q\left(t,r\right) = -2\pi rT\frac{\partial\phi\left(t,r\right)}{\partial r}$$

met T [L^2/T] het doorlaatvermogen.

De r komt alleen voor in $h\left(u,r/R\right)$. Differentiatie naar r daarvan levert

$$\frac{\partial h(u,r/R)}{\partial r} = \frac{u}{R}\frac{Y_0(u)J_1(\frac{r}{R}u) - J_0(u)Y_1(\frac{r}{R}u)}{J_0^2(u) + Y_0^2(u)}$$

waarin met $\frac{dJ_{0(x)}}{dx} = -J_1\left(x\right)$ en $\frac{dY_0(x)}{dx} = -Y_1\left(x\right)$

Het debiet kan dan als volgt worden bepaald

$$\begin{aligned}
Q\left(t,r\right) &= -2\pi rT\frac{\partial\phi(t,r)}{\partial r}\\
&= -(2\pi rT)(-h\frac{2}{\pi})\int_{u=0}^{\infty}\frac{1}{u}\frac{\partial h(u,r/R)}{\partial r}\exp\left(-\frac{u^2t}{\beta^2R^2}\right)du\\
&= h\,4rT\int_{u=0}^{\infty}\frac{1}{u}\frac{\partial h(u,r/R)}{\partial r}\exp\left(-\frac{u^2t}{\beta^2R^2}\right)du\\
&= h\,4rT\int_{u=0}^{\infty}\frac{1}{u}\frac{u}{R}\frac{Y_0(u)J_1(\frac{r}{R}u) - J_0(u)Y_1(\frac{r}{R}u)}{J_0^2(u) + Y_0^2(u)}\exp\left(-\frac{u^2t}{\beta^2R^2}\right)du\\
&= 4Th\frac{r}{R}\int_{u=0}^{\infty}\frac{Y_0(u)J_1(\frac{r}{R}u) - J_0(u)Y_1(\frac{r}{R}u)}{J_0^2(u) + Y_0^2(u)}\exp\left(-\frac{u^2t}{\beta^2R^2}\right)du\\
&= 4Th\frac{r}{R}\int_{z=-\infty}^{+\infty}u\frac{Y_0(u)J_1(\frac{r}{R}u) - J_0(u)Y_1(\frac{r}{R}u)}{J_0^2(u) + Y_0^2(u)}\exp\left(-\frac{u^2t}{\beta^2R^2}\right)dz, \quad u = e^z
\end{aligned}$$

In de laatste regel ga ik integreren langs de logaritmische schaal door te schrijven$d\ln u = dz = \frac{du}{u}$ zodat $du = u\,dz$, vandaar de u voor de uitdrukking met de Besselfuncties. Uiteraard geldt in de laatste regel $u = \exp(z)$

Dus voor het debiet op de rand van de cilinder zet $r = R$, wat de uitdrukking iets eenvoudiger maakt. Voor het berekenen moeten we de integraal uitrekenen; of dat lukt hangt deels af van het gedrag van de uitdrukking onder de integraal. Hetzelde geldt uiteraard voor het uitrekenen van de integraal in de uitdrukking van $\phi(t, r)$. Figuur 1 geeft het verloop van het argument onder de integraal van ϕ voor 4 verschillende waarden van r/R. Uiteraard is dat argument idendiek aan nu voor $r/R = 1$. Voor waarden an $r/R > 1$ is het argument ongelijk aan nul en heeft de te oscillaties die uit de besselfuncties voortkomen.

Figuur 2 geeft het argument onder de ingegraal voor de uitdrukking van het debiet. Voor dezelfde uitgangspunten en voor waarden voor r/R als figuur 1. Ook hier krijgen we de te verwachten oscillaties, althans voor waarden van $r/R > 1$. Hoe groter r/R hoe meer oscilaties optreden en hoe moeilijker de (numerieke) integratie zal zijn.

Echter, het verloop van het argument van Q voor $r = R$, dus precies op de rand van de cilinder, heeft geen enkele oscillatie en zou daarom zonder problemen integreerbaar moeten zijn. Dus zou het berekenen van $Q(t, r = R)$ probleemloos moeten verlopen. De integratie verloopt ook verder probleemloos, echter de uitkomst klopt niet zoals blijkt uit figuur 3.

Het is duidelijk dat de uitkomst niet kan kloppen want de grafieken van ϕ moeten voor elke $r > R$ op nul beginnen en asymptotisch naar de eindwaarde (1) lopen. De grafieken voor Q moeten voor elke $r > R$ op 0 beginnen en dan aansluiten op die voor $r = R$.

Tenslotte de juiste uitkomst, die heel veel verschilt van die van Bruggeman zoals ik die berekend heb. In figuur 4 staan de resultaten die ik heb berekend met Modflow, met mijn eigen fdm3t, met ttim, met mijn benadering met convolutie en, tenslotte, in oranje, wat ik kreeg met de formule van Bruggeman voor ϕ en de daaruit afgeleide formule van Q, beide voor $r = R$. De met Bruggeeman berekende stijghoogte voor $r = R$ is uiteraard goed (1), maar triviaal. De met de afgeleide formule voor het debiet berekende debiet wijkt echter enorm af van de juiste waarde die overtuigend met de 4 andere methoden is berekend. Ik heb geen idee hoe dat komt.

Hieronder nog even hoe ik de formules van Bruggeman bereken. Ik gebruik daarvoor een paar uiterst korte functies waarvan **argFi** en **argQ** het argument berekent onder de integraal en **Fint** en **Qint** de integraal zelf numeriek berekent met **scipy.integrate.quad, h** berekent de functie **h(u, r/R)** en **dhdr** de afgeleide daarvan naar r**.**

Listing
```
# Solution of Bruggeman 220_03, flow from cylinder of given R after sudden head change at t=0.
import numpy as np
import matplotlib.pyplot as plt
from scipy.special import j0 as J0, y0 as Y0, j1 as J1, y1 as Y1
from scipy.integrate import quad

import settings

def h(u=None, r=None, R=None, **_):
    rho = r / R
    return (J0(u) * Y0(u * rho) - Y0(u) * J0(u * rho)) / (J0(u) ** 2 + Y0(u) ** 2)

def dhdr(u=None, r=None, R=None, **_):
    """Return partial h/ partial r."""
    rho = r / R
    return u / R * ((Y0(u) * J1(rho * u) - J0(u) * Y1(rho * u))) / (J0(u) ** 2 + Y0(u) ** 2)

def argFi(z=None, r=None, R=None, beta=None, t=None, **_):
    u = np.exp(z)
    return h(u, r, R) * np.exp(- (u / (beta * R)) ** 2 * t)
```

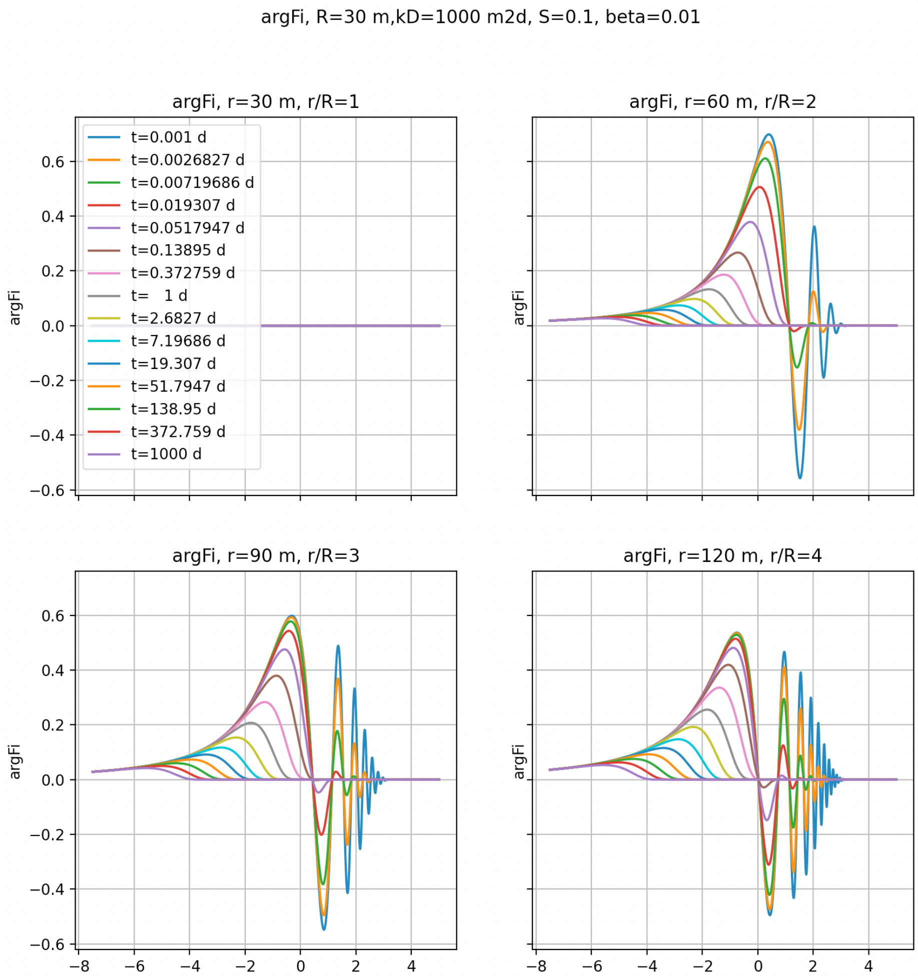

Figure 1: Argument onder de integraal van $\phi(t, r)$

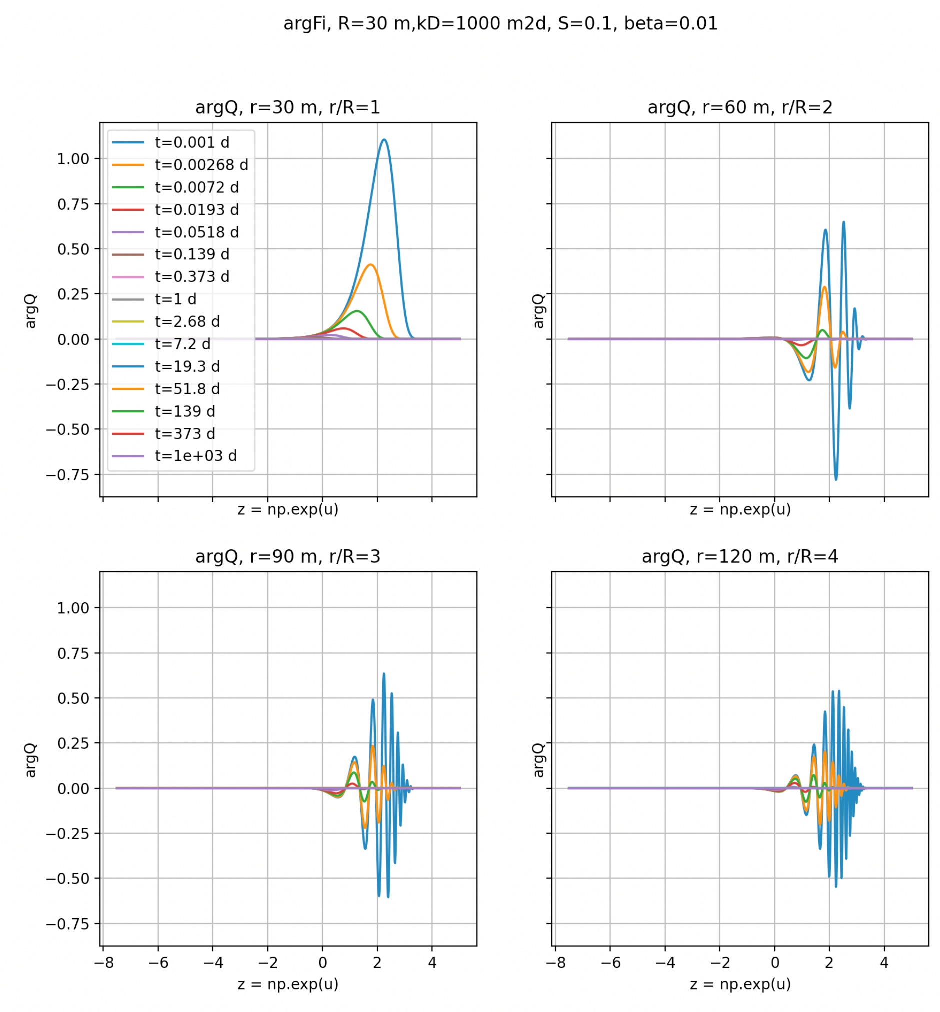

Figure 2: Argument onder de integraal van $Q\left(t,r\right)$

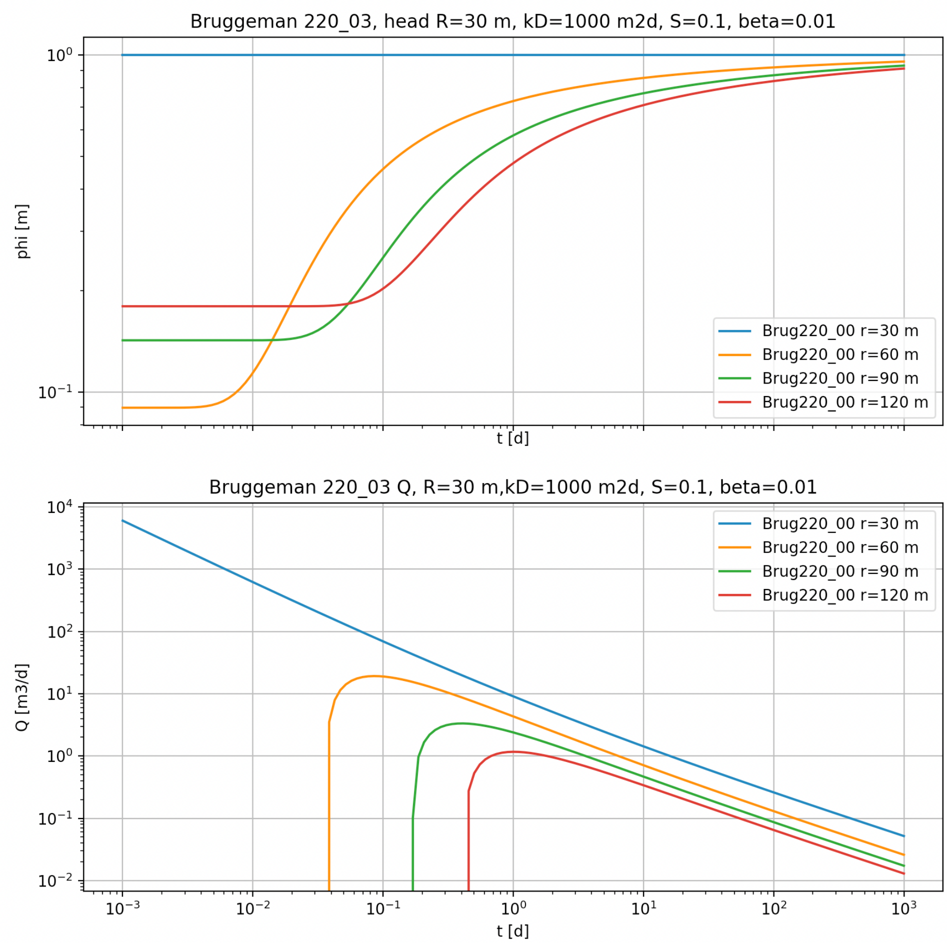

Figure 3: Berekende $\phi\left(t,r\right)$ en $Q\left(t,r\right)$

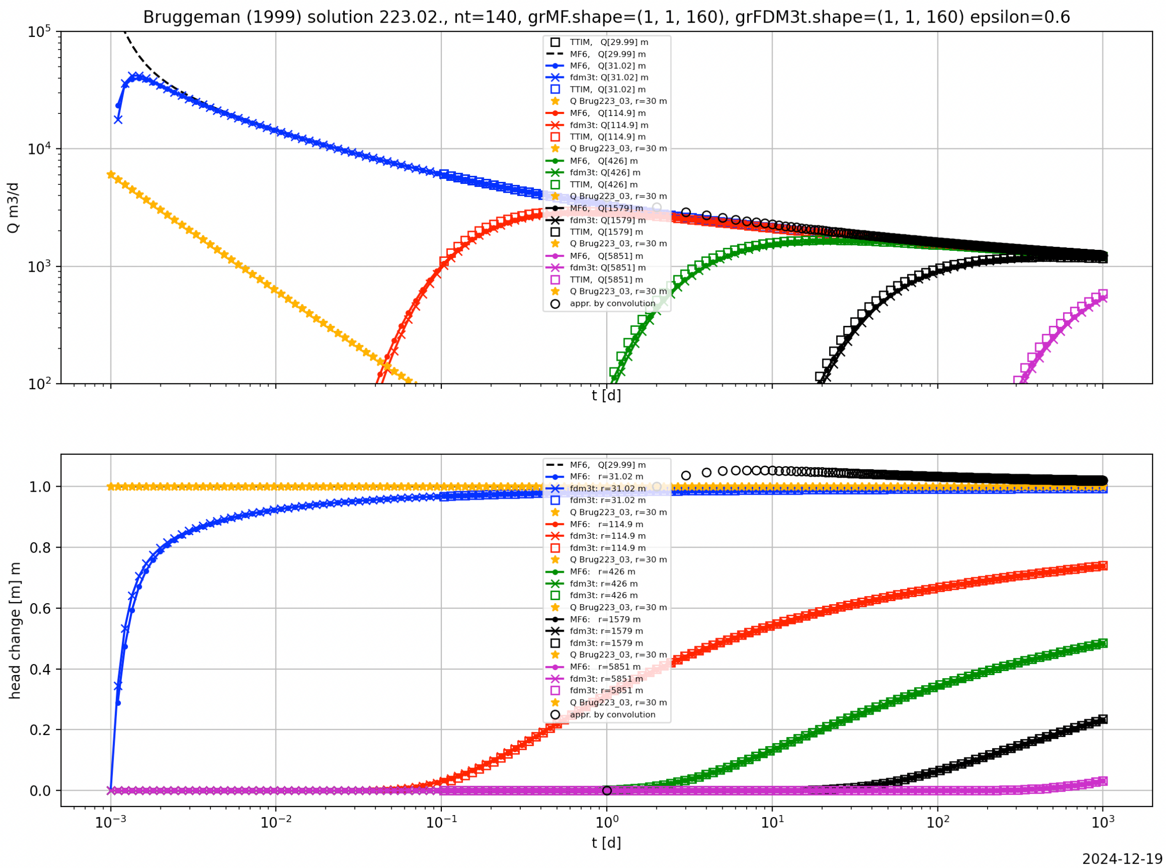

Figure 4: Uitkomsten van: Modflow, fdm3t, TTIM, benaderd met convolutie en analytische Bruggeman 223_02.

```python
def Fint(a, b, args=None):
    """Return phi(t, r).
    Parameters
    ----------
    a, b: limit of integration (should be -inf, inf)
    args: tuple of floats of remaining parameters.
        args = (r, R, beta, t)
    """
    return 1. - 2. / np.pi * quad(argFi, a, b, args=args, limit=200)[0]

def argQ(z=None, r=None, R=None, beta=None, t=None, **kw):
    u = np.exp(z)
    return u * dhdr(u, r, R) * np.exp(- (u / (beta * R)) ** 2 * t)

def Qint(a, b, args=None):
    """Return Q(t, r).
    Parameters
    ----------
    a, b: floats, limits of the integration
    args: tuple of parameters
        args = (r, R, beta, t)
    """
    return quad(argQ, a, b, args=args, limit=200)[0]


kD = np.sum(settings.props['kr'] * settings.props['D'])
S  = np.sum(settings.props['ss'] * settings.props['D'])
beta = np.sqrt(S / kD)

z = np.linspace(-7.5, 5., 1001)

R = settings.props['r'][1]
rs = np.array([1.0, 2.0, 3.0, 4.0]) * R

figsize = (10., 10.)

times = settings.props['t']

fig, axs = plt.subplots(2, 2, sharex=True, sharey=True, figsize=figsize)
fig.suptitle(f"argFi, R={R:.4g} m,kD={kD:.4g} m2d, S={S:.4g}, beta={beta:.4g}")

for r, ax in zip(rs, axs.ravel()):
    ax.set_title(f"argFi, r={r:.4g} m, r/R={r/R:.4g}")
    ax.set_ylabel('argFi')
    ax.grid(True)

    for t in times[::10]:
        ax.plot(z, argFi(z=z, r=r, R=R, beta=beta, t=t), label=f't={t:4g} d')
    if np.isclose(r, 30):
        ax.legend(loc='upper left')


fig, axs = plt.subplots(2, 2, sharex=True, sharey=True, figsize=figsize)
fig.suptitle(f"argFi, R={R:.4g} m,kD={kD:.4g} m2d, S={S:.4g}, beta={beta:.4g}")
for r, ax in zip(rs, axs.ravel()):
    ax.set_title(f"argQ, r={r:.4g} m, r/R={r/R:.4g}")
    ax.set_ylabel('argQ')
    ax.set_xlabel('z = np.exp(u)')
```

```python
    ax.grid(True)

    for t in times[::10]:
        ax.plot(z, argQ( z=z, r=r, R=R, beta=beta, t=t), label=f't={t:.3g} d')
    if np.isclose(r, 30):
        ax.legend(loc='upper left')

_, (ax1, ax2) = plt.subplots(2, 1, sharex=True, figsize=figsize)
ax1.set_title(f"Bruggeman 220_03, head R={R:.4g} m, kD={kD:.4g} m2d, S={S:.4g}, beta={beta:.4g}")
ax1.set_xlabel('t [d]')
ax1.set_ylabel('phi [m]')
ax1.set_yscale('log')
ax1.set_xscale('log')
ax1.grid(True)

ax2.set_title(f"Bruggeman 220_03 Q, R={R:.4g} m,kD={kD:.4g} m2d, S={S:.4g}, beta={beta:.4g}")
ax2.set_xlabel('t [d]')
ax2.set_ylabel('Q [m3/d]')
ax2.set_yscale('log')
ax2.set_xscale('log')
ax2.grid(True)

# Dit stuk is wat ingewikkeld omdat ik de uitkomst als een dictonary in een ander bestand
# importeer om deze daar over de grafiek van de numerieke uitkomsten te tekenen.
B = dict()
B['Q'] = dict()
B['F'] = dict()
B['t'] = dict()
F = dict()
a, b = -7.5, 5
for r in rs:
    ri = np.round(r)
    B['Q'][ri] = np.zeros_like(times)
    B['F'][ri] = np.zeros_like(times)
    B['t'][ri] = np.zeros_like(times)
    for it, t in enumerate(times):
        args = (r, R, beta, t)
        B['Q'][ri][it] = 4 * kD * settings.hb * Qint(a, b, args=args)
        B['F'][ri][it] = settings.hb * Fint(a, b, args=args)
        B['t'][ri][it] = t
    ax1.plot(times, B['F'][ri], label=f'Brug220_00 r={r:.4g} m')
    ax2.plot(times, B['Q'][ri], label=f'Brug220_00 r={r:.4g} m')

ax1.legend()
ax2.legend()

#plt.show()
```