Including source terms in channel flow

Hansjörg Seybold[1]

October 12, 2009

1 Including a body force in the channelflow spectral DNS code

1.1 Modifying the Time steppign algorithms

1.1.1 CNAB Style Scheme

We start with equation (51) of the channel flow manual, where now $\hat{f}(x,t)$ is the spectral transformed of the forcing term $f(x,t)$.

$$\frac{\partial \tilde{\mathbf{u}}}{\partial t} + \tilde{\nabla}\tilde{q} = \tilde{\mathbf{L}}\tilde{\mathbf{u}} - \widetilde{\mathbf{N}(\mathbf{u})} + \tilde{C} + \tilde{f} \tag{1}$$

In the CNAB the function is evaluated at the intermediate timestep $n+1/2$ where (52)-(56) are the discretizationsfor the linear and nonlinear term. The body force term does not have to be approximated but can be evaluated directly at time $n+1/2$ thus adding just $\tilde{f}^{n+1/2}$ to equation (57)

$$\left[\frac{1}{\Delta t} - \frac{\tilde{\mathbf{L}}}{2}\right]\tilde{\mathbf{u}}^{n+1} + \tilde{\nabla}\tilde{q}^{n+1} =$$
$$\left[\frac{1}{\Delta t} + \frac{\tilde{\mathbf{L}}}{2}\right]\tilde{\mathbf{u}}^{n} + \tilde{\nabla}\tilde{q}^{n} + \frac{3}{2}\widetilde{N}^{n} - \frac{1}{2}\widetilde{N}^{n-1} + \frac{1}{2}\tilde{C}^{n+1} + \frac{1}{2}\tilde{C}^{n} + \tilde{f}^{n+1/2} \tag{2}$$

Comparing the coefficients with the general form and deviding by β_i yields

$$\tilde{\mathbf{R}} = \left[\frac{1}{\beta_i\Delta t} + \frac{\alpha_i\tilde{\mathbf{L}}}{\beta_i}\right]\tilde{\mathbf{u}}^{n} + \frac{\alpha_i}{\beta_i}\tilde{\nabla}\tilde{q}^{n} + \frac{\gamma_i}{\beta_i}\widetilde{N}^{n} + \frac{\zeta_i}{\beta_i}\widetilde{N}^{n-1} + \tilde{C}^{n+1} + \frac{\alpha_i}{\beta_i}\tilde{C}^{n} + \frac{1}{\beta_i}\tilde{f}^{n+1/2} \tag{3}$$

The code has to be modified as follows: First the body force has to be calculated at time $t_{n+1/2}$.

```
  case CNAB2:
...
c_[0]      = 0.5;
break;
...
//=====================================================================//
// create a force field
FlowField force(un.Nx(), un.Ny(), un.Nz(), 3, un.Lx(),
un.Lz(), un.a(), un.b(), Physical, Physical);
// fill the force field with the force values in spectral space
// the force function can be directly evaluated at the intermediate timestep
// n+1/2
setBodyForce(t_+ c_[j]*dt_, force);
//=====================================================================//
```

Then the RHS of the tau solver has to be modidied as follows:

```
for (int ny=0; ny<Nyd_; ++ny) {
          Rxk_.add(ny, c*uk_[ny] - Dx*Pk_[ny]
                   // body force--------------------------
                   +1/beta_[j] * force.cmplx(mx,ny,mz,0)//
                   //-----------------------------------
                   - g_b*fj_.cmplx(mx,ny,mz,0) - z_b*fj1_.cmplx(mx,ny,mz,0));
          Ryk_.add(ny, c*vk_[ny] - Pyk_[ny]
                   // body force--------------------------
                   +1/beta_[j] * force.cmplx(mx,ny,mz,1)//
                   //-----------------------------------
                   - g_b*fj_.cmplx(mx,ny,mz,1) - z_b*fj1_.cmplx(mx,ny,mz,1));
          Rzk_.add(ny, c*wk_[ny] - Dz*Pk_[ny]
                    // body force-------------------------
                   +1/beta_[j] * force.cmplx(mx,ny,mz,2)//
                    //----------------------------------
                   - g_b*fj_.cmplx(mx,ny,mz,2) - z_b*fj1_.cmplx(mx,ny,mz,2));
       }
```

1.1.2 SMRK2 Scheme

The SMRK scheme is a low storage Runge Kutta scheme originally proposed by Wray (A. Wray, Minimal Storage Time-advancement Schemes for Spectral Methods, NASA Ames Research Center, 1990.)
The scheme is implemented as described in Spalart, Moser & Rogers, JCP 96(1990) with an implicit treatment of the linear part.

$$\begin{aligned}
\mathbf{u}' &= \mathbf{u}_n + \Delta t\left(\tilde{\mathbf{L}}(\alpha_1\mathbf{u}_n + \beta_1\mathbf{u}') + \gamma_1\widetilde{N}^n\right) && (4)\\
\mathbf{u}'' &= \mathbf{u}' + \Delta t\left(\tilde{\mathbf{L}}(\alpha_2\mathbf{u}' + \beta_2\mathbf{u}'') + \gamma_2\widetilde{N}' + \zeta_2\widetilde{N}^n\right) && (5)\\
\mathbf{u}_{n+1} &= \mathbf{u}'' + \Delta t\left(\tilde{\mathbf{L}}(\alpha_3\mathbf{u}'' + \beta_3\mathbf{u}''') + \gamma_3\widetilde{N}'' + \zeta_3\widetilde{N}^n\right) && (6)
\end{aligned}$$

where the coefficients are given by

$$\alpha_1 = \frac{29}{96} \quad \alpha_2 = -\frac{3}{40} \quad \alpha_3 = \frac{1}{6} \quad \beta_1 = \frac{37}{160} \quad \beta_2 = \frac{5}{24} \quad \beta_3 = \frac{1}{6} \qquad (7)$$

$$\gamma_1 = \frac{8}{15} \quad \gamma_2 = \frac{5}{12} \quad \gamma_3 = \frac{3}{4} \quad \zeta_1 = 0 \quad \zeta_2 = -\frac{17}{60} \quad \zeta_3 = \frac{5}{12} \qquad (8)$$

In the NASA report Center for Modeling of Turbulence and Transition (Research Briefs 1991) the scheme of Wray is described in the following form

$$\begin{aligned}
\mathbf{u}' &= \mathbf{u}_n + \Delta t\frac{8}{15}R_n && (9)\\
\mathbf{u}'' &= \mathbf{u}' + \Delta t\left(\frac{5}{12}R_n - \frac{17}{60}R'\right) = \mathbf{u}_n + \Delta t\left(\frac{1}{4}R_n + \frac{5}{12}R'\right) && (10)\\
\mathbf{u}_n &= \mathbf{u}'' + \Delta t\left(\frac{3}{4}R_n - \frac{5}{12}R''\right) = \mathbf{u}_n + \Delta t\left(\frac{1}{4}R_n + \frac{3}{4}R''\right) && (11)
\end{aligned}$$

where R' is the RHS evaluated at point $\mathbf{u}'$ etc. With the condition for maximal consistency order $c_i = \sum_{j=1}^{i-1} a_{ij}$ follows:

$$c_1 = 0 \qquad c_2 = \frac{8}{15} \qquad c_3 = \frac{2}{3} \qquad (12)$$

These coefficients were also mentioned as the time conefficients of the scheme of Wray in Kennedy Carpenter Appl. Num. Math. 35(2000). Thus the scheme in the general form reads as follows:

$$\mathbf{u}' = \mathbf{u}_n + \frac{8}{15}\Delta t\,(g_n + f(t_n)) \tag{13}$$

$$\mathbf{u}'' = \mathbf{u}' + \frac{5}{12}\Delta t\,(g_n + f(t_n)) - \frac{17}{60}\Delta t\left(g(\mathbf{u}') + f(t_n + \frac{8}{15}\Delta t)\right) \tag{14}$$

$$\mathbf{u}_{n+1} = \mathbf{u}' + \frac{3}{4}\,(g_n + f(t_n)) - \frac{5}{12}\Delta t\left(g(\mathbf{u}'') + f(t_n + \frac{2}{3}\Delta t)\right) \tag{15}$$

finally the body force term can be included in the scheme of Spalart, Moser & Rogers, where the linear term is treated implicitly, but we do not have to care about this modified coefficients.

$$\mathbf{u}' = \mathbf{u}_n + \Delta t\left(\tilde{\mathbf{L}}(\alpha_1\mathbf{u}_n + \beta_1\mathbf{u}') + \gamma_1\left[\widetilde{N}^n + f(t_n)\right]\right) \tag{16}$$

$$\begin{aligned}\mathbf{u}'' = \mathbf{u}' + \Delta t\Big(&\tilde{\mathbf{L}}(\alpha_2\mathbf{u}' + \beta_2\mathbf{u}'')\\ &+\gamma_2\left[\widetilde{N}' + f(t_n + \frac{5}{12}\Delta t)\right] + \zeta_2\left[\widetilde{N}^n + f(t_n)\right]\Big)\end{aligned} \tag{17}$$

$$\begin{aligned}\mathbf{u}_{n+1} = \mathbf{u}'' + \Delta t\Big(&\tilde{\mathbf{L}}(\alpha_3\mathbf{u}'' + \beta_3\mathbf{u}''')\\ &+\gamma_3\left[\widetilde{N}'' + f(t_n + \frac{2}{3}\Delta t)\right] + \zeta_3\left[\widetilde{N}^n + f(t_n)\right]\Big)\end{aligned} \tag{18}$$

Implementation:
Extend the coefficients in CNABstyleDNS in DNS.h with

```
array<Real> c_;        // time coefficients for integration
```

Modify the coefficients in CNABstyleDNS::CNABstyleDNS file dns.cpp

```
case SMRK2:
...
c_[0]     = 0.0;             c_[1] = 8.0/15.0;         c_[2] = 2.0/3.0;
```

The body force then is added with the same coefficients as the nonlinear terms evaluated at the intermediate time steps.
Due to the fact that after introducing the body force terms the SMRK2 scheme and the CNAB2 scheme do not fit with the coefficients anymore the following changes have been applied to conduct the calculation in the CNABstyle Class:

1. Two more coefficients are introduced to describe the body force contribution in dns.h

   ```
   array<Real> i_b1_;     // coefficient for the body force term
   array<Real> i_b2_;     // coefficient for the body force term
   ```

   where i_b1 stands for the intermediate time coefficients and i_b2 are the contributions from the previous time step.

2. when the algorithms are set the coefficients are chosen such that the correct force term contributions are added.

   ```
   case CNAB2:
     i_b1_[0]    =1.0;  i_b2_[0]    =0.0;
   case SMRK2
     i_b1_[0]  = gamma_[0];    i_b1_[1] = gamma_[1];    i_b1_[2] = gamma_[2];
     i_b2_[0]  = zeta_[0];     i_b2_[1] = zeta_[1];     i_b2_[2] = zeta_[2];
   ```

3. The body force then is calcculated in two parts to save storage for the old body force field;

   (a) first add the intermediate time contributions

```
// create a force field
FlowField force(un.Nx(), un.Ny(), un.Nz(), 3, un.Lx(),
un.Lz(), un.a(), un.b(), Physical, Physical);
// fill the force field with the force values in spectral space
setBodyForce(t_+ c_[j]*dt_, force);
for (int ny=0; ny<Nyd_; ++ny) {
  Rxk_.add(ny, c*uk_[ny] - Dx*Pk_[ny]
  // body force--------------------
  +i_b1_[j]/beta_[j] * force.cmplx(mx,ny,mz,0)//
  //------------------------------------
  - g_b*fj_.cmplx(mx,ny,mz,0) - z_b*fj1_.cmplx(mx,ny,mz,0));
  Ryk_.add(ny, c*vk_[ny] - Pyk_[ny]
  // body force--------------------
  +i_b1_[j]/beta_[j] * force.cmplx(mx,ny,mz,1)//
  //------------------------------------
  - g_b*fj_.cmplx(mx,ny,mz,1) - z_b*fj1_.cmplx(mx,ny,mz,1));
  Rzk_.add(ny, c*wk_[ny] - Dz*Pk_[ny]
  // body force-------------------
  +i_b1_[j]/beta_[j] * force.cmplx(mx,ny,mz,2)//
  //------------------------------------
  - g_b*fj_.cmplx(mx,ny,mz,2) - z_b*fj1_.cmplx(mx,ny,mz,2));
}
```

   (b) Then add the body force contributions of the previous timestep

```
// this is the second part of the force term in the SMRK2
setBodyForce(t_, force);
 for (int ny=0; ny<Nyd_; ++ny) {
  Rxk_.add(ny,i_b2_[j]/beta_[j] * force.cmplx(mx,ny,mz,0));
  Ryk_.add(ny,i_b2_[j]/beta_[j] * force.cmplx(mx,ny,mz,1));
  Rzk_.add(ny,i_b2_[j]/beta_[j] * force.cmplx(mx,ny,mz,2));
}
```

1.1.3 Multistep Schemes

The Multistep scheme BDF4 reads like

$$\begin{aligned}\frac{\eta}{\Delta t}\tilde{\mathbf{u}}^{n+1} &= -\frac{1}{\Delta t}\left[a_1\tilde{\mathbf{u}}^{n} + a_2\tilde{\mathbf{u}}^{n-1} + a_3\tilde{\mathbf{u}}^{n-2} + a_4\tilde{\mathbf{u}}^{n-3}\right] \\ &\quad + \left[b_1\tilde{\mathbf{F}}^{n} + b_2\tilde{\mathbf{F}}^{n-1} + b_3\tilde{\mathbf{F}}^{n-2} + b_4\tilde{\mathbf{F}}^{n-3}\right] \qquad (19)\end{aligned}$$

where $\tilde{\mathbf{F}}^k$ is the RHS of the k-th step.

$$\tilde{\mathbf{F}}^k = -\tilde{\nabla}\tilde{q}^k + \mu\tilde{\mathbf{L}}\tilde{\mathbf{u}}^k - \tilde{N}^k + \tilde{C}^k + \tilde{f}^k \qquad (20)$$

Thus the Body force can be just included by adding the value of the force term to the RHS.
Alternatively one can include the body force directly at timestep $n+1$. which then yields:

$$\begin{aligned}\frac{\eta}{\Delta t}\tilde{\mathbf{u}}^{n+1} &= -\frac{1}{\Delta t}\left[a_1\tilde{\mathbf{u}}^{n} + a_2\tilde{\mathbf{u}}^{n-1} + a_3\tilde{\mathbf{u}}^{n-2} + a_4\tilde{\mathbf{u}}^{n-3}\right] \\ &\quad + \left[b_1\tilde{\mathbf{F}}^{n} + b_2\tilde{\mathbf{F}}^{n-1} + b_3\tilde{\mathbf{F}}^{n-2} + b_4\tilde{\mathbf{F}}^{n-3}\right] + \tilde{f}^{n+1} \qquad (21)\end{aligned}$$

Here $\tilde{\mathbf{F}}^k$ is given by the term $\tilde{\mathbf{F}}^k = -\tilde{\nabla}\tilde{q}^k + \mu\tilde{\mathbf{L}}\tilde{\mathbf{u}}^k - \tilde{N}^k + \tilde{C}^k$. The body force term in (21) is considered in the same way as the linear term in Canuto p.131 eq.(4.80).
The first scheme (19) makes directly use of the explicit extrapolation of the body force and time $t + dt$ first scheme. The following modifications have to be applied to the timestepping:

1. Calculation of the body force at time t_n

```
//==========================================================================//
// create a force field
FlowField force(un.Nx(), un.Ny(), un.Nz(), 3, un.Lx(),
un.Lz(), un.a(), un.b(), Physical, Physical);
// fill the force field with the force values in spectral space at the given
// timestep
setBodyForce(t_, force);
//==========================================================================//
```

2. Then add the force to the RHS in f_[0]

```
f_[0]+=force;
```

Alternatively Eq.(21) can be used where the force term is directly evaluated at time $t+dt$:

```
//==========================================================================//
// create a force field
FlowField force(un.Nx(), un.Ny(), un.Nz(), 3, un.Lx(),
un.Lz(), un.a(), un.b(), Physical, Physical);
// fill the force field with the force values in spectral space
// the force function can be directly evaluated at the final timestep
// n+1
setBodyForce(t_+dt_, force);
//==========================================================================//
```

is added to the RHS directly

```
//==========================================================================//
for (int ny=0; ny<Nyd_; ++ny) {
  Rxk_.add(ny, force.cmplx(mx,ny,mz,0));
  Ryk_.add(ny, force.cmplx(mx,ny,mz,1));
  Rzk_.add(ny, force.cmplx(mx,ny,mz,2));
}
//==========================================================================//
```

1.1.4 RKCN Schemes

The RKCN scheme used in channelflow follows the low storage RK3 scheme introduced by Wiliamson (1980). The scheme is described in Peyret S.149 and Canuto S.108, where the temporal evaluation of the RHS is only described in Canuto.
In a general form the scheme reads like this

$$u_0 = u^n \tag{22}$$

$$Q_1 = \Delta t g(u_0, t_n) \qquad u_1 = u_0 + \frac{1}{3}Q_1 \tag{23}$$

$$Q_2 = -\frac{5}{9}Q_1 + \Delta t g(u_1, t + \frac{\Delta t}{3}) \qquad u_2 = u_1 + \frac{15}{16}Q_2 \tag{24}$$

$$Q_3 = -\frac{153}{128}Q_2 + \Delta t g(u_2, t + \frac{3\Delta t}{4}) \qquad u_3 = u_2 + \frac{1}{8}Q_3 = u^{n+1} \tag{25}$$

for a arbitrary differential equation fo the form $y' = g(y,t)$. Thus as the body force term can be evaluated independantly at any given time t one obtains with the unchanged implicit discretization of the linear term

$$u_0 = u^n \tag{26}$$
$$Q_1 = \Delta t N(u_0, t_n) + \Delta t f(t_n, x) \tag{27}$$
$$u_1 = u_0 + \frac{1}{3}Q_1 + \frac{\Delta t}{6}\left(L(u_0) + L(u_1)\right) \tag{28}$$
$$Q_2 = -\frac{5}{9}Q_1 + \Delta t N(u_1) + \Delta t f(t_n + \frac{\Delta t}{3}, x) \tag{29}$$
$$u_2 = u_1 + \frac{15}{16}Q_2 + \frac{5\Delta t}{24}\left(L(u_1) + L(u_2)\right) \tag{30}$$
$$Q_3 = -\frac{153}{128}Q_2 + \Delta t N(u_2) + \Delta t f(t_n + \frac{3\Delta t}{4}, x) \tag{31}$$
$$u_3 = u_2 + \frac{1}{8}Q_3 + \frac{\Delta t}{8}\left(L(u_0) + L(u_1)\right) = u^{n+1} \tag{32}$$

which implies that the body force term has to be added when calculating the Q_i.
To include the body force in the RKCN2 scheme the inner substep loop has to be adapted. The time coefficients are stored in the variable $D\_[i]$ where $D\_[0] = 0$, $D\_[1] = 1/3$, $D\_[2] = 3/4$. The variable D has been added in the header file *dns.h*, class *RungeKuttaDNS*

```
array<Real> D_;  // coefficient for the force term
```

This yields the following modification of the RK scheme

```
RungeKuttaDNS::RungeKuttaDNS(const FlowField& u, const ChebyCoeff& Ubase,
                             Real nu, Real dt, const DNSFlags& flags, Real t)
{
  ...
  switch (algorithm) {
    case CNRK2:
    D_[0] = 0.0;    D_[1] = 1.0/3.0;   D_[2] = 3.0/4.0;     // Canuto !!!
}
...
void RungeKuttaDNS::advance(FlowField& un, FlowField& qn, int Nsteps) {
  for (int j=0; j<Nsubsteps_; ++j) {

    FlowField& uj(un); // Store uj in un during substeps, reflect in notation

    //==================================================================//
    FlowField force(un.Nx(), un.Ny(), un.Nz(), 3, un.Lx(),
    un.Lz(), un.a(), un.b(), Physical, Physical);
    // fill the force field with the force values in spectral space
    // the force function can be directly evaluated at the intermediate timestep
    setBodyForce(t_+D_[j]*dt_, force);
    //==================================================================//
    // Efficient implementation of
    // Q_{j+1} = A_j Q_j + N(u_j)}   where N = -u grad u
    // Q_{j+1} = A_j Q_j - f(u_j)}   where f =  u grad u
    // Q_j = Q_{j+1}
    Qj_ *= A_[j];
    navierstokesNL(uj, ubase_, Ubase_, Qj1_, tmp_, tmp2_, flags_.nonlinearity);
    Qj_ -= Qj1_; // subtract because navierstokesNL(u) = u grad u = -N(u)
    Qj_+=force;
  ...
}
```

2 Implementing the force term

- The forcing function should be specified in the top level code

```
class BodyForce
{
   public:
   bool operator()(Real t, FlowField &Forcefield) const
   {
     ...
   }
};
```

- The forcing function can be passed as a function object to the DNS constructor.
- two alternatives in the DNS constructor: with and without force. Allocate memory for the force field only if the force term exists and set a flag to identify the existence of a force.
- the field bodyforce is part of the DNS.

```
DNS::DNS(const FlowField& u,
            Real nu,Real dt, const DNSFlags& flags, Real t);
DNS::DNS(const FlowField& u, const  BodyForce &F,
            Real nu, Real dt, const DNSFlags& flags, Real t);
{
  // DNS creates a local field force of the right size with resize
  // otherwise the force field remains empty.
}
```

- make a if statement in the advance function which excludes the force calculation at the beginning of the advance step, such that the if is only executed once in an advance step.

3 Testing the force term

3.1 Poiseulle flow

Test a force term by imposing a constant force in x-direction. This should reproduce the parabolic velocity profile of a poiselle flow equal to a constant pressure drop.

```
force(nx,ny,nz,0) = 2.0*1./40.0;
```

As test function the routine dnsParabolaTest.cpp has been adjusted

- Re Number 40
- zero base field $\mathbf{U}(y)$ and zero fluctuation velocity $\mathbf{u}_n(x)$.
- constant presssure drop $\mathrm{d}p/\mathrm{d}x = -2\nu$

After a while the velocity profile adapts to the parabolic profile; If no pressure drop is assigned the flow field remains constant zero.

```
flags.dPdx = 0.0;//-2.0*nu;     // mean streamwise pressure gradient.
```

Now the pressure drop is set to zero and the body force is switched on. The different integration schemes have to be tested seperately. We start with the one step routines as these are needed for initialization of the multistep schemes.

1. Test of the CNRK2:
2. Test of the SMRK2:
3. Test of the CNAB2 with SMRK2 initialization:
4. Test of the Multistep algorithms BDF3:

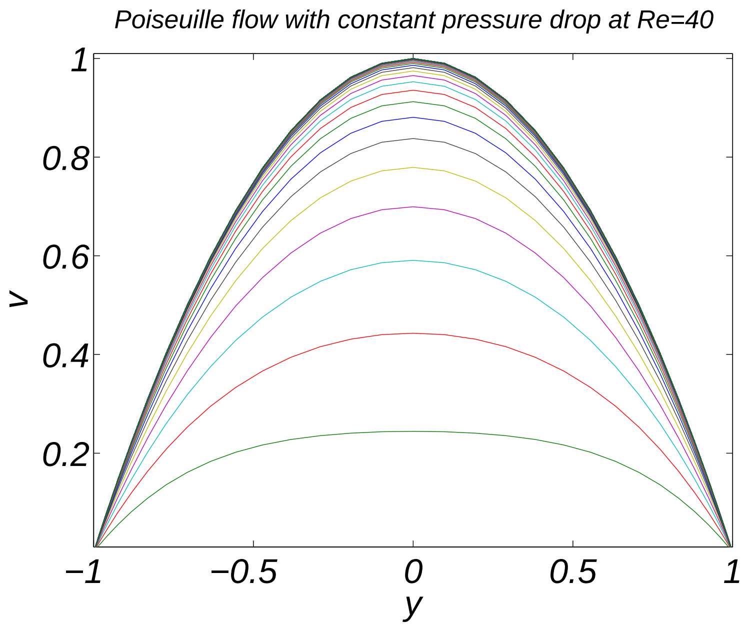

Figure 1: Development of the parabolic velocity profile. 500 steps, CNAB2, $CLF \leq 0.3$

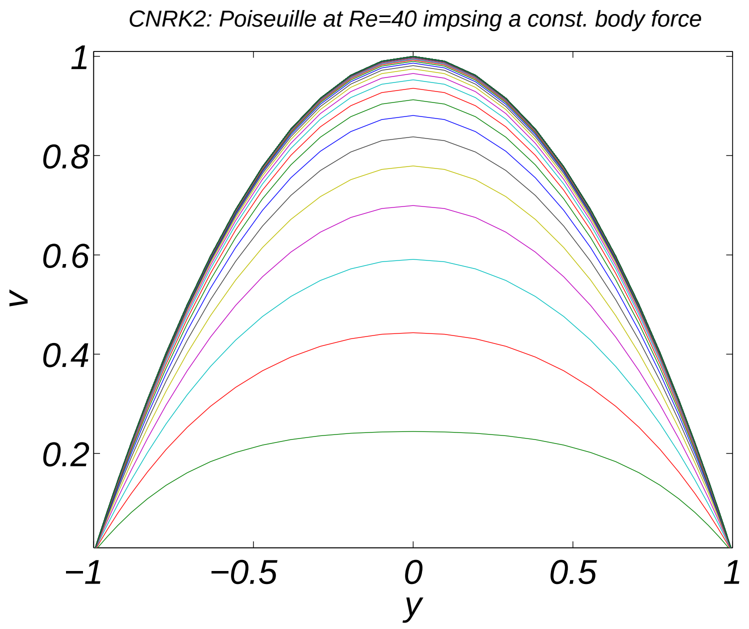

Figure 2: Development of the parabolic velocity profile at Re=40. 500 steps, CNRK2, $CLF \leq 0.3$, constant body force $f = 2\nu$

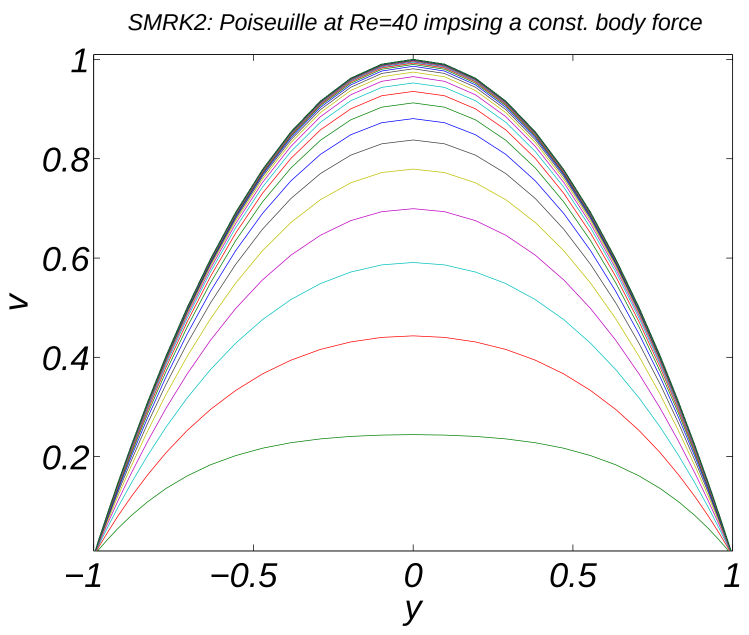

Figure 3: Development of the parabolic velocity profile at Re=40. 500 steps, SMRK2, $CLF \leq 0.3$, constant body force $f = 2\nu$

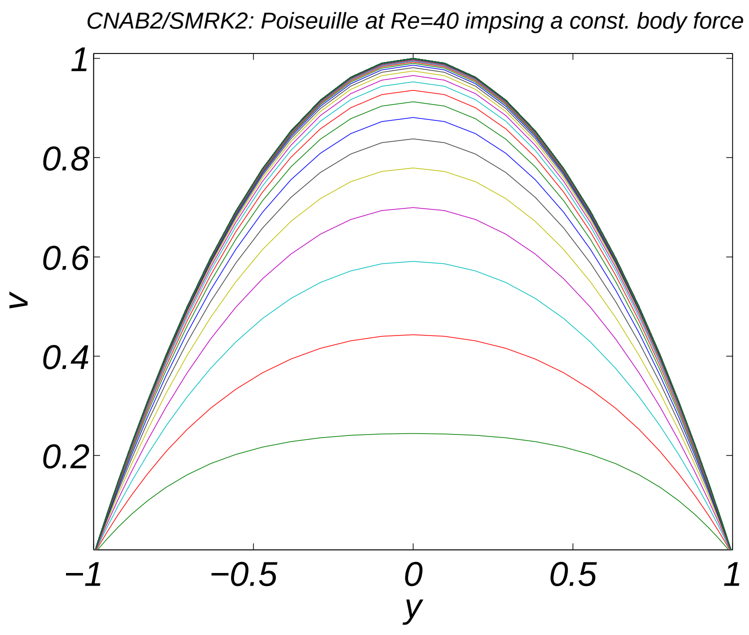

Figure 4: Development of the parabolic velocity profile at Re=40. 500 steps, CNAB2, $CLF \leq 0.3$, constant body force $f = 2\nu$

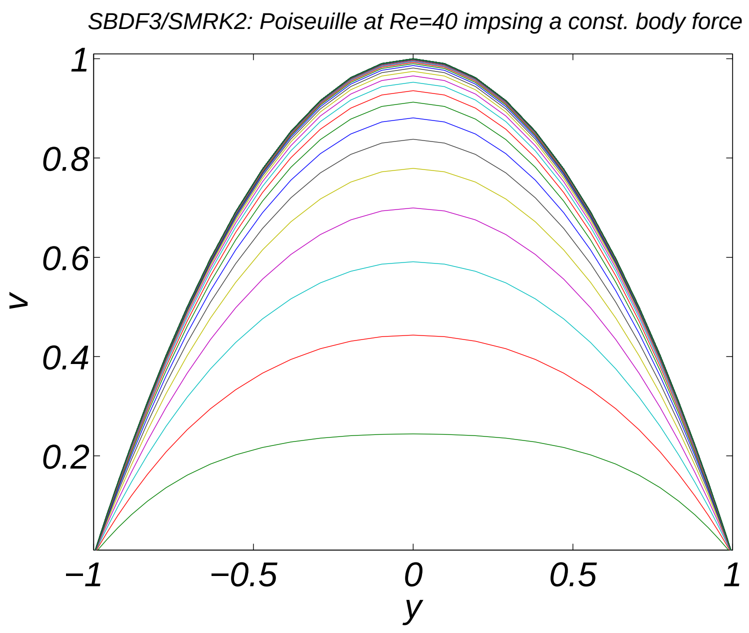

Figure 5: Development of the parabolic velocity profile at Re=40. 500 steps, SBDF3, $CLF \leq 0.3$, constant body force $f = 2\nu$

3.2 Stokes flow

The Navier-Stokes equations including body forces are given by

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\nabla\mathbf{u} = -\nabla p + \frac{1}{Re}\Delta\mathbf{u} + f \tag{33}$$

By imposing a velocity field $\mathbf{u}(\mathbf{x})$ that fits the boundary conditions and is divergence free, one can calculate the body force which is necessary to obtain $\mathbf{u}(\mathbf{x})$ as a solution of the Navier-Stokes equations.

$$f = \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\nabla\mathbf{u} + \nabla p - \frac{1}{Re}\Delta\mathbf{u} \tag{34}$$

The body force obtained in this way can be plugged into the numerical solver to obtain the numerical solution fo the velocity field $\hat{\mathbf{u}}(\mathbf{x})$
Let $k_x = k_z$ and $L_x = L_z$ then the field:

$$\mathbf{u} = \cos(\omega t)\cdot\left(1 - y^2\right)\begin{pmatrix} \sin\frac{2\pi kx}{L}\cos\frac{2\pi kz}{L} \\ 0 \\ -\cos\frac{2\pi kx}{L}\sin\frac{2\pi kz}{L} \end{pmatrix} \tag{35}$$

is divergence free and fits the boundary conditions.
Proof: As $\operatorname{div} F = \partial_x F + \partial_y F + \partial_z F$ the divergence of (35) is given by

$$\partial_x \mathbf{u} = \frac{2\pi k}{L}\cos\frac{2\pi kx}{L}\cos\frac{2\pi kz}{L}\cdot\left(1 - y^2\right) \tag{36}$$

$$\partial_y \mathbf{u} = 0 \tag{37}$$

$$\partial_z \mathbf{u} = \frac{2\pi k}{L}\cos\frac{2\pi kx}{L}\cos\frac{2\pi kz}{L}\cdot\left(1 - y^2\right) \tag{38}$$

Furthermore the boundary conditions are:

$$\mathbf{u}(x=0)=0, \qquad \mathbf{u}(x=L)=0 \tag{39}$$
$$\mathbf{u}(y=-1)=0 \qquad \mathbf{u}(y=1)=0 \tag{40}$$
$$\mathbf{u}(z=0)=\mathbf{u}(y=L) \tag{41}$$

Plugging Eq. 35 into the Navier-Stokes equations yields:

1. Difusion term:

$$\Delta\mathbf{u} = \underbrace{\nabla(\nabla\mathbf{u})}_{=0 \text{ as } \nabla\mathbf{u}=0} - \nabla\times(\nabla\times\mathbf{u}) \tag{42}$$
$$= 2\cos(\omega t)\left[\left(\frac{2\pi k}{L}\right)^2\left(1-y^2\right)+1\right]\begin{pmatrix} -\sin\frac{2\pi kx}{L}\cos\frac{2\pi kz}{L} \\ 0 \\ \sin\frac{2\pi kz}{L}\cos\frac{2\pi kx}{L}\end{pmatrix} \tag{43}$$

2. Advection term:

$$\mathbf{u}\nabla\mathbf{u} = \frac{2\pi k}{L}\cos^2(\omega t)(y^2-1)^2\begin{pmatrix} \sin\frac{2\pi kx}{L}\cos\frac{2\pi kx}{L} \\ 0 \\ \sin\frac{2\pi kz}{L}\cos\frac{2\pi kz}{L}\end{pmatrix} \tag{44}$$

3. Time derivative:

$$\frac{\partial\mathbf{u}}{\partial t} = \omega\sin(\omega t)\cdot\left(1-y^2\right)\begin{pmatrix} -\sin\frac{2\pi kx}{L}\cos\frac{2\pi kz}{L} \\ 0 \\ \cos\frac{2\pi kx}{L}\sin\frac{2\pi kz}{L}\end{pmatrix} \tag{45}$$

Altogether we obtain for the body force in the case $\nabla p = 0$

$$f = \left(\omega(1-y^2)\sin\omega t - \frac{2}{Re}\cos\omega t\left[\left(\frac{2\pi k}{L}\right)^2(1-y^2)+1\right]\right)\begin{pmatrix} -\sin\frac{2\pi kx}{L}\cos\frac{2\pi kz}{L} \\ 0 \\ \cos\frac{2\pi kx}{L}\sin\frac{2\pi kz}{L}\end{pmatrix}$$
$$+\frac{2\pi k}{L}\cos^2(\omega t)(y^2-1)^2\begin{pmatrix} \sin\frac{2\pi kx}{L}\cos\frac{2\pi kx}{L} \\ 0 \\ \sin\frac{2\pi kz}{L}\cos\frac{2\pi kz}{L}\end{pmatrix} \tag{46}$$

3.2.1 static case: $\omega = 0$

1. Case CNAB:

   Visually compared the two solutions look very similar

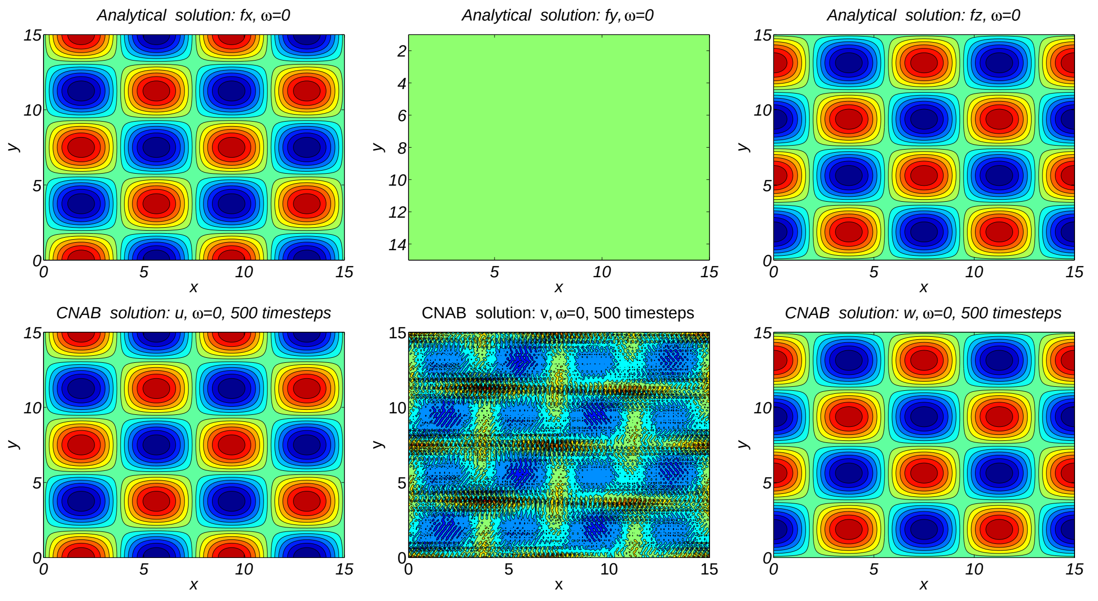

Figure 6: Cuts through the analytical flow field which fits the boundary conditions (top) and the CNAB solution for the different velocity variable (bottom)

The maximal error for the u-velocity in the $y = 0$ plane is 5×10^{-7}, a plot of the error in the cut plane is shown in Fig. 7.

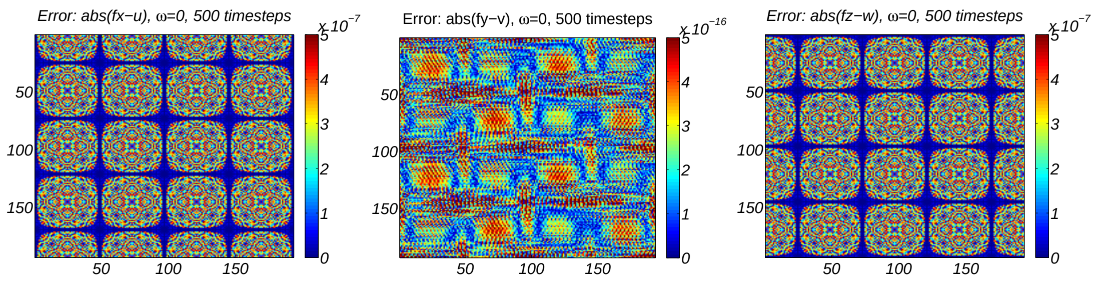

Figure 7: Error abs($f - u$) in the velocity of the numerical solution of the CNAB algorithm in the plane $y = 0$. Note that the range of the colorbars.

2. Case SBDF: For the SBDF algorithm we obtain the samoe results as in the case of the CNAB integration.

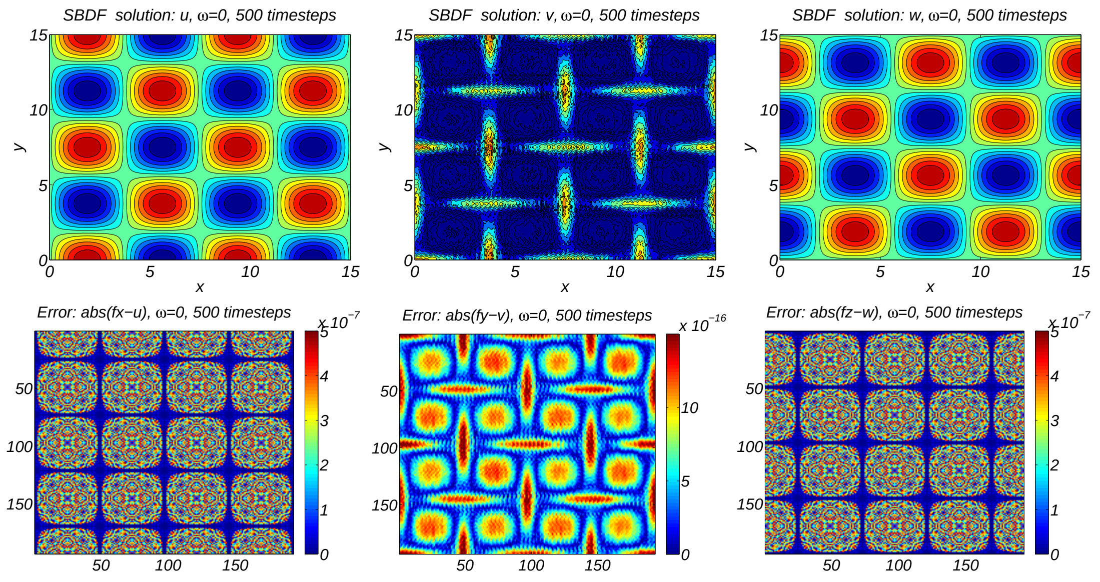

Figure 8: Cuts through the SBDF solution for the different velocity variable (top) and the corresponding error compared to the analytical solution

3. Case SMRK: The SMRK integration algorithm has a equivalent behavior as the previous ones. The values for the v velocity are in the range of 1^{-15}

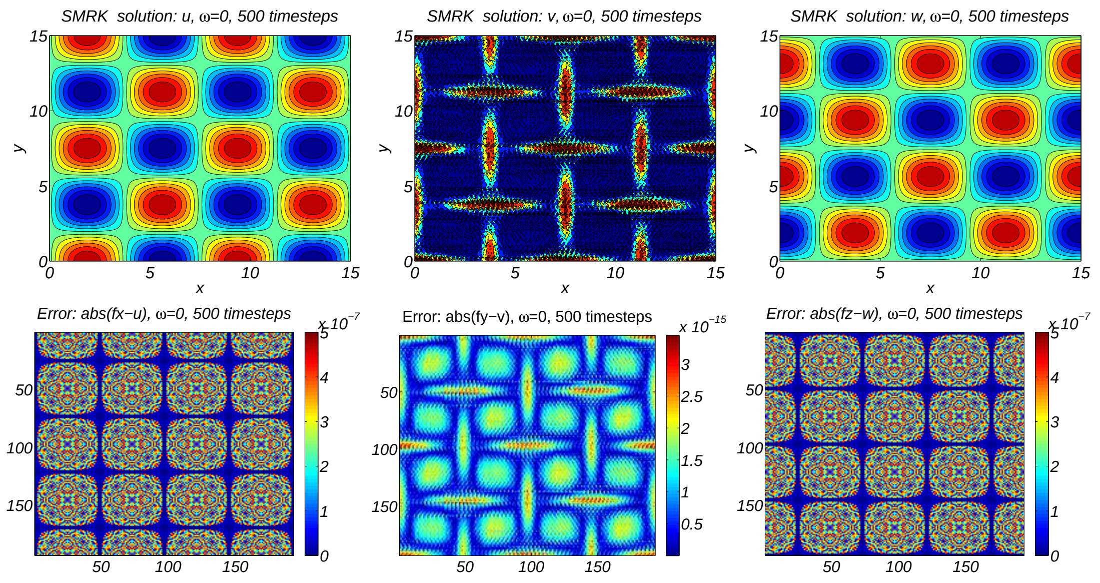

Figure 9: Cuts through the SMRK solution for the different velocity variable (top) and the corresponding error compared to the analytical solution

4. Case CNRK: The CNRK integration algorithm has a equivalent behavior as the previous ones. The values for the v velocity are in the range of 1^{-15}

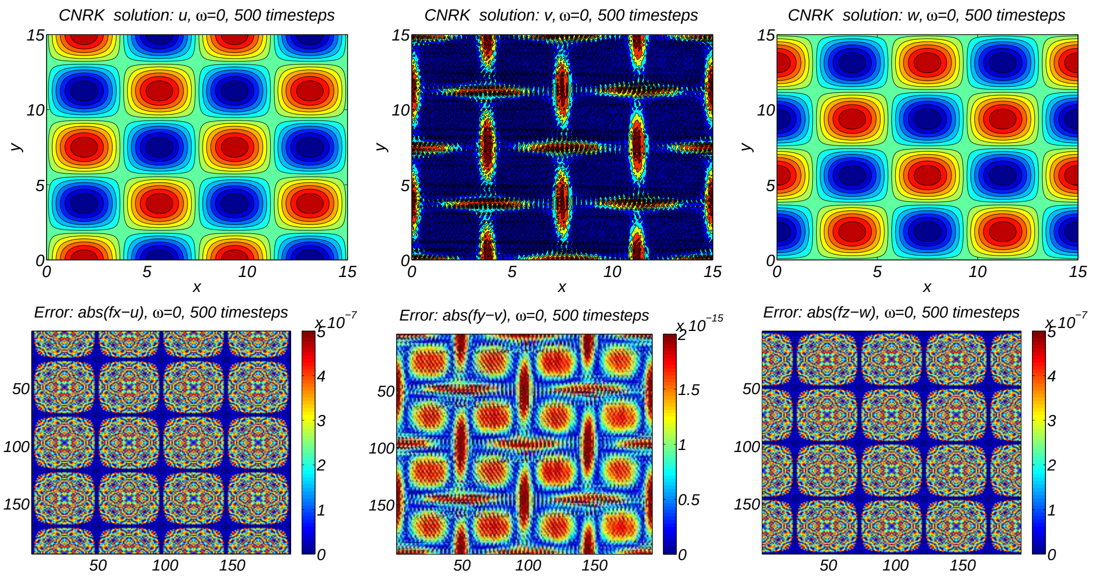

Figure 10: Cuts through the CNRK solution for the different velocity variable (top) and the corresponding error compared to the analytical solution