Assignment 3: The Discrete Geometric Conservation Law

- The assignment should be handed in at the latest Wednesday 27th March 2024 12:00 in order to receive full credit, otherwise a 2 point penalty is applied to the grade of the assignment.
- The assigment can be made in groups of **at most 3 students**.
- Please be sure to include the names and student numbers in your report.
- Please hand in an electronic version (also a scanned version of a (clearly) hand-written report is accepted) through Brightspace (only one submission per group is required).
- You do not have to make a full report (including summary, list of figures, list of symbols etc), but indicate clearly the question numbers when giving your answer / results / discussion of results.
- When an implementation is requested, please provide (only) the code snippets where you made your implementation.

Introduction

In fluid-structure interaction simulations, the fluid equations are solved on a deforming mesh. An often used method for dealing with deforming meshes is the Arbitrary Lagrangian-Eulerian (ALE) approach, which allows the fluid dynamics equations to be solved on arbitrarily moving/deforming control volumes. When using the ALE approach, a Geometric Conservation Law (GCL) can be identified that ties the mesh deformation to mesh velocities, used in the governing equations. In this exercise we will investigate the GCL for a range of simple test problems.

Governing equations

A general formulation for the fluid dynamics equations on a domain Ω is

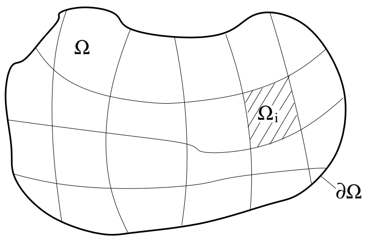

Figure 1: Computational domain

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega}\mathbf{U}\mathrm{d}V + \oint_{\partial\Omega}\left[\mathbf{F}(\mathbf{U}) - \mathbf{U}\frac{\mathrm{d}\vec{x}}{\mathrm{d}t}\right]\cdot\vec{n}\mathrm{d}S = 0, \tag{1}$$

wherein $\mathbf{U}$ the conservative variables $(\rho, \rho\vec{u}, \rho E)$, $\mathbf{F}$ the flux vector containing convective and diffusive terms, $\frac{\mathrm{d}\vec{x}}{\mathrm{d}t}$ the mesh velocity, $\vec{n}$ the outward pointing normal to surface $\partial\Omega$. Suppose we discretize the domain Ω using a finite number of volumes Ω_i, we can write for a single volume

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_i}\mathbf{U}\mathrm{d}V + \oint_{\partial\Omega_i}\left[\mathbf{F}(\mathbf{U}) - \mathbf{U}\frac{\mathrm{d}\vec{x}}{\mathrm{d}t}\right]\cdot\vec{n}\mathrm{d}S = 0. \tag{2}$$

This will be the starting point for deriving the continuous Geometric Conservation Law, and the semi-discrete and discrete versions.

1 Geometric Conservation Law

One of the viewpoints on the significance of the GCL is that it ensures uniform flow to remain uniform on arbitrarily moving meshes. This feature is used to derive first of all the continous GCL.

Continuous GCL

In the derivation we assume $\mathbf{U} = \mathbf{U}_0$ to be uniform in the domain Ω and hence one may expect that $\mathbf{F}_0 = \mathbf{F}(\mathbf{U}_0)$ also uniform. The control volume Ω_i has a closed surface $\partial\Omega_i$.

Exercise 1.1 Derive the GCL by substitution of $\mathbf{U}_0$ into (2)

Note that up till now, the relation still uses continuous functions and the continuous GCL is independent of the discretization method used for numerically solving the fluid dynamics equations. For certain methods, such as space-time discretizations, the GCL is generally satisfied in a natural way. For other discretization approaches, such as finite-volume discretizations, this may not be the case. Since a major part of engineering CFD codes use the finite volume approach, we continue in a finite-volume framework for obtaining the semi-discrete formulation of the GCL.

Semi-discrete GCL

First the control volume Ω_i is discretized to V_i and its boundary $\partial\Omega_i$ to a sommation of discrete faces (or edges in 2D), as shown in Fig. 2. The spatial integrals are discretized using

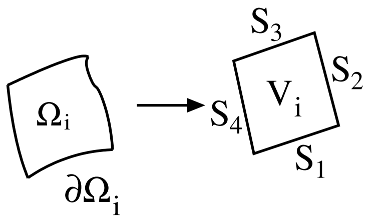

Figure 2: Discrete domain of a single control volume V_i.

$$\oint_{\partial\Omega_i} \frac{\mathrm{d}\vec{x}}{\mathrm{d}t} \cdot \vec{n}\mathrm{d}S \approx \sum_{faces} \left(\frac{\mathrm{d}\vec{x}}{\mathrm{d}t} \cdot \vec{n}S \right)_{face}, \tag{3}$$

with S the face surface area for each disrete face, so that we obtain the semi-discrete GCL

$$\frac{\mathrm{d}}{\mathrm{d}t} V_i = \sum_{faces} \left(\frac{\mathrm{d}\vec{x}}{\mathrm{d}t} \cdot \vec{n}S \right)_{face}. \tag{4}$$

Exercise 1.2 What do the left and right sides of (4) represent (what is their physical interpretation)?

Discrete GCL

In (4) we still have a temporal derivative. To obtain a fully discrete equation a wide variety of time integration schemes can be chosen to discretize the temporal derivative. Here we will consider four implicit integration schemes:

- **Backward Euler** The first order Backward Euler scheme is quite popular due to its simplicity and stability. The integration of a quantity ϕ yields

$$\frac{\phi^{n+1}-\phi^{n}}{\Delta t}+R(\phi^{n+1})=0, \tag{5}$$

wherein $R(\phi^{n+1})$ is the spatial discretization, evaluated at the implicit time level t_{n+1}.

- **BDF2** The second order Backward Differencing scheme is often used in CFD codes. It obtains a second order accuracy by using the state of an additional time level t_{n-1}. The integration of a quantity ϕ yields

$$\frac{\alpha_1\phi^{n+1}+\alpha_2\phi^{n}+\alpha_3\phi^{n-1}}{\Delta t}+R(\phi^{n+1})=0, \tag{6}$$

wherein $\vec{\alpha}=(\frac{3}{2},-2,\frac{1}{2})$ for BDF2 and $R(\phi^{n+1})$ is the spatial discretization, evaluated at the implicit time level t_{n+1}. Since the method needs data at time level t_{n-1}, which is generally unavailable at $t_n=t_0$, the first time step is generally performed with a Backward Euler scheme for which $\vec{\alpha}=(1,-1,0)$.

- **θ-scheme** The θ-scheme is a combination between a Backward Euler ($\theta=1$) and Forward Euler ($\theta=0$) integration scheme. It obtains a second order accuracy when $\theta=\frac{1}{2}$. The integration of a quantity ϕ yields

$$\frac{\phi^{n+1}-\phi^{n}}{\Delta t}+\theta R(\phi^{n+1})+(1-\theta)R(\phi^{n})=0, \tag{7}$$

wherein $R(\phi^{n+1})$ is the spatial discretization, evaluated at the implicit time level t_{n+1} and $R(\phi^n)$ is the spatial discretization, evaluated at the explicit time level t_n.

- **ESDIRK** The ESDIRK schemes are a class of multi-stage Runge-Kutta schemes. The general formulation for an s-stage scheme is denoted by

$$\begin{aligned}\frac{\phi^{(k)}-\phi^{n}}{\Delta t}+\sum_{j=1}^{k}a_{kj}R(\phi^{(j)}) &= 0, \\ \frac{\phi^{n+1}-\phi^{n}}{\Delta t}+\sum_{j=1}^{s}b_{j}R(\phi^{(j)}) &= 0\end{aligned} \tag{8}$$

for which the coefficients a and b are generally written as a Butcher-tableau. For example, a 4-stage ESDIRK scheme has the following Butcher-tableau

$$\begin{array}{c|cccc} c_1 & 0 & 0 & 0 & 0 \\ c_2 & a_{21} & \gamma & 0 & 0 \\ c_3 & a_{31} & a_{32} & \gamma & 0 \\ c_4 & a_{41} & a_{42} & a_{43} & \gamma \\ \hline & b_1 & b_2 & b_3 & b_4 \end{array}$$

The abbreviation ESDIRK stands for Explicit first stage ($a_{11}=0$), Single diagonal coefficent ($a_{ii}=\gamma$), Diagonally Implicit ($a_{ij}=0, \forall j>i$) Runge-Kutta scheme, which is reflected in the Butcher tableau. The method may also use $b_i=a_{si}$ in which case the solution of the final stage is already the solution at t_{n+1} ($\phi^{n+1}=\phi^{(s)}$).

As shown in the lecture, the change in volume of the cell is decomposed into volume changes caused by the motion of each cell face individually:

$$V^{n+1} - V^n = \sum_{faces} \Delta V_{face}^{n+1}, \tag{9}$$

wherein ΔV_{face}^{n+1} the so-called swept-volume for a face moving from its location at t_n to its location at t_{n+1}. Therefore, we obtain for the Backward Euler scheme the following Discrete-GCL (D-GCL)

$$\frac{V^{n+1} - V^n}{\Delta t} = \sum_{faces} \frac{\Delta V_{face}^{n+1}}{\Delta t} = \sum_{faces} \left(\frac{\mathrm{d}\vec{x}}{\mathrm{d}t} \cdot \vec{n} S \right)_{face}^{n+1}, \tag{10}$$

which is enforced for each face individually

$$\frac{\Delta V_{face}^{n+1}}{\Delta t} = \left(\frac{\mathrm{d}\vec{x}}{\mathrm{d}t} \cdot \vec{n} S \right)_{face}^{n+1}, \tag{11}$$

so that the face normal velocity $(\frac{\mathrm{d}\vec{x}}{\mathrm{d}t} \cdot \vec{n})^{n+1}$ should be

$$(\frac{\mathrm{d}\vec{x}}{\mathrm{d}t} \cdot \vec{n})^{n+1} = \frac{\Delta V_{face}^{n+1}}{\Delta t S}, \tag{12}$$

in order to ensure that the D-GCL is satisfied.

Exercise 1.3	Derive the D-GCL for the BDF2 scheme
Exercise 1.4	Derive the D-GCL for the ESDIRK-scheme

2 Simulation of uniform flow on a one-dimensional moving mesh

Next we investigate the simulation of uniform flow on a moving mesh and the disturbances that can be generated when not satisfying the D-GCL. The matlab files necessary are stored in folder `matlab\DGCL-1D\`. We start with the example shown in the lectures for a one-dimensional testcase (Fig. 3) for which an exact mesh motion is known:

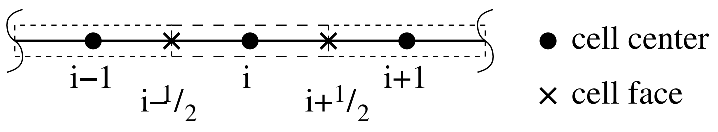

Figure 3: One dimensional mesh.

$$\begin{aligned} x &= x_0 + \Delta x_0 \sin(2\pi x_0)\sin(2\pi t), && (13)\\ \frac{\mathrm{d}x}{\mathrm{d}t} &= 2\pi \Delta x_0 \sin(2\pi x_0)\cos(2\pi t), && (14)\end{aligned}$$

wherein $x_0 = x(t = 0)$ and $\Delta x_0 = 1/N$, with N the number of finite volume cells to discretize the domain [0,1]. The model that is discretized in given in (2). We assume a constant scalar field u, so that $F(u)$ is constant as well and can be eliminated from the equation. As long as u remains uniform, this is correct. However, for the sake of simplicity, we do not add the influence of the numerical flux F to the equation when the solution does not remain uniform (when the D-GCL is violated). The model is therefore

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{\Omega_i} u \mathrm{d}V - \oint_{\partial\Omega_i} \left[u \frac{\mathrm{d}x}{\mathrm{d}t} \right] \cdot n \mathrm{d}S = 0, \tag{15}$$

for which the boundary integral reduces to a sommation of a left and right face contribution:

$$\frac{\mathrm{d}}{\mathrm{d}t}(uV)_i + \left[u\frac{\mathrm{d}x}{\mathrm{d}t}\right]_{i-1/2} - \left[u\frac{\mathrm{d}x}{\mathrm{d}t}\right]_{i+1/2} = 0, \tag{16}$$

wherein we use a simple avarage to approximate the solution at the cell faces $i \pm 1/2$:

$$u_{i+1/2} = \frac{u_i + u_{i+1}}{2}. \tag{17}$$

When e.g. a Backward Euler time integration is chosen, (16) and (17) result in

$$\frac{u_i^{n+1}V_i^{n+1} - u_i^n V_i^n}{\Delta t} + \frac{u_{i-1}^{n+1} + u_i^{n+1}}{2}\left(\frac{\mathrm{d}x}{\mathrm{d}t}\right)_{i-1/2}^{n+1} - \frac{u_i^{n+1} + u_{i+1}^{n+1}}{2}\left(\frac{\mathrm{d}x}{\mathrm{d}t}\right)_{i+1/2}^{n+1} = 0, \tag{18}$$

or,

$$\begin{aligned} &\left[V_i^{n+1} + \frac{\Delta t}{2}\left(\frac{\mathrm{d}x}{\mathrm{d}t}\right)_{i-1/2}^{n+1} - \frac{\Delta t}{2}\left(\frac{\mathrm{d}x}{\mathrm{d}t}\right)_{i+1/2}^{n+1}\right] u_i^{n+1} \\ &\qquad + \left[\frac{\Delta t}{2}\left(\frac{\mathrm{d}x}{\mathrm{d}t}\right)_{i-1/2}^{n+1}\right] u_{i-1}^{n+1} - \left[\frac{\Delta t}{2}\left(\frac{\mathrm{d}x}{\mathrm{d}t}\right)_{i+1/2}^{n+1}\right] u_{i+1}^{n+1} = V_i^n u_i^n, \end{aligned} \tag{19}$$

which results in a system

$$L\mathbf{u}^{n+1} = R\mathbf{u}^n. \tag{20}$$

The matlab program `dgcl_BE.m` sets-up this system of equations and solves the system for every time step. The influence of the DGCL is through the $(\frac{\mathrm{d}x}{\mathrm{d}t})$ terms in L.

Backward Euler For the Backward Euler scheme, the swept volumes of the faces reduce to a change in position of the face coordinate χ and therefore its D-GCL (12) becomes

$$\left(\frac{\mathrm{d}x}{\mathrm{d}t}\right)^{n+1} = \frac{\chi^{n+1} - \chi^n}{\Delta t}, \tag{21}$$

which is implemented into the matlab file `dgcl_BE.m`, along with the options to choose the exact mesh velocities at $t_{n+1/2}$ and t_{n+1}.

Exercise 2.1 In `dgcl_BE.m` we set `N = 20`, `tend = 10` and for each `method=1,2,3` we set the time step and plot format according to Table 1. When switching from one method to the other, you can close the figures to start a clean plot. You can verify that satisfying the D-GCL preserves the constant solution. As a rough estimate for the order with which the error reduces with respect to Δt, one can judge the amplitude decrease in the solution at t_{end} for various Δt, e.g. when the amplitude $\approx$ halves when Δt is halved, the error reduces with $O(\Delta t)$. Which orders do you find approximately for `method=1,2`?

Table 1: Time step and formst settings.

case	Δt	`fmt`
1	0.1	'-b'
2	0.05	'-r'
3	0.025	'-g'

We are going to make a more thorough analysis of the error introduced by not satisfying the D-GCL by considering two errors:

- The root-mean-square error at the final time,
- The root-mean-square error over the whole simulation.

The first error is defined by

$$\epsilon_1 = \sqrt{\frac{\sum_{i=1}^{N}(u_i^M - u_{ex}(x_i, t_M))^2}{N}}, \tag{22}$$

wherein u_{ex} the exact solution, N the total number of unknowns in space and M the total number of time steps taken. The second error is defined by

$$\epsilon_2 = \sqrt{\frac{\sum_{j=1}^{M}\sum_{i=1}^{N}(u_i^j - u_{ex}(x_i, t_j))^2}{NM}}. \tag{23}$$

We perform a time step refinement study by running the simulation at time steps $\Delta t = 0.1 \cdot 2^{-k}, k = 0, 1, 2, 3, 4, 5, 6$. Note that by setting `show_sol = 0` in the Matlab script, one can speed up the computation since the intermediate results are not shown. For each of the chosen time steps, determine the errors ϵ_1 and ϵ_2 for `method=1,2,3`. By visualizing the ϵ versus Δt results in a log-log plot, one can determine the order with which the error decreases as Δt decreases by measuring the slope of the curve in the plot. When the time steps are small enough and the method is consistent, the error should behave like

$$\epsilon \approx c\Delta t^p, \tag{24}$$

when the error decreases with $O(\Delta t^p)$. Therefore, taking the logarithm on both sides yields

$$\log \epsilon \approx \log c + p \log \Delta t, \tag{25}$$

which in a log-log plot is similar to a linear function $Y = C + pX$ (with $Y = \log \epsilon$ and $X = \log \Delta t$), with a slope of p.

Exercise 2.2 In `dgcl_BE.m` implement the computation of the errors ϵ_1 and ϵ_2; set `N = 20`, `tend = 10` and for each `method=1,2,3` determine the error for the time steps $\Delta t = 0.1 \cdot 2^{-k}, k = 0, 1, 2, 3, 4, 5, 6$ and visualize the results in a log-log plot.
From the plot determine the order of consistency with which the errors tend to zero when the time step is reduced.
Even though the error should be zero when satisfying the D-GCL one may still find an error unequal to zero - what kind of error is observed here?

As a second step we will analyse the error by making a Taylor series expansion around the location in time for which the exact velocity is prescribed in `method = 1,2`; as you know the Taylor series expansion about t_n for a small perturbation δ yields

$$x(t_n + \delta) = x(t_n) + \delta \left.\frac{\mathrm{d}x}{\mathrm{d}t}\right|_{t_n} + \frac{\delta^2}{2}\left.\frac{\mathrm{d}^2x}{\mathrm{d}t^2}\right|_{t_n} + \frac{\delta^3}{6}\left.\frac{\mathrm{d}^3x}{\mathrm{d}t^3}\right|_{t_n} + \frac{\delta^4}{24}\left.\frac{\mathrm{d}^4x}{\mathrm{d}t^4}\right|_{t_n} + O(\delta^5), \tag{26}$$

for a polynomial approximation of degree 4. Notice that the time derivatives of x are taken at the time instance t_n. We know how to satisfy the D-GCL for the Backward Euler time integration scheme: use Eq. (21). As the coordinated χ of the faces are in fact spatial locations in x, we can also write, for a given face location at x, that satisfying the D-GCL yields:

$$\left(\frac{\mathrm{d}x}{\mathrm{d}t}\right)^{n+1} = \frac{x^{n+1} - x^n}{\Delta t}. \tag{27}$$

Therefore when using the exact mesh velocity at t_{n+1}, instead of the D-GCL, the difference between the two (the error) is

$$\epsilon = \frac{x^{n+1} - x^n}{\Delta t} - \left.\frac{\mathrm{d}x}{\mathrm{d}t}\right|_{t_{n+1}}. \tag{28}$$

We can now use a Taylor expansion around t_{n+1} to express x^n as

$$x^n = x^{n+1} - \Delta t \left.\frac{\mathrm{d}x}{\mathrm{d}t}\right|_{t_{n+1}} + \frac{\Delta t^2}{2}\left.\frac{\mathrm{d}^2x}{\mathrm{d}t^2}\right|_{t_{n+1}} - \frac{\Delta t^3}{6}\left.\frac{\mathrm{d}^3x}{\mathrm{d}t^3}\right|_{t_{n+1}} + \frac{\Delta t^4}{24}\left.\frac{\mathrm{d}^4x}{\mathrm{d}t^4}\right|_{t_{n+1}} + O(\Delta t^5), \tag{29}$$

which, when substituted into (28) provides the approximation of the error

$$\epsilon = \left[\left.\frac{\mathrm{d}x}{\mathrm{d}t}\right|_{t_{n+1}} - \left.\frac{\mathrm{d}x}{\mathrm{d}t}\right|_{t_{n+1}}\right] - \frac{\Delta t}{2}\left.\frac{\mathrm{d}^2x}{\mathrm{d}t^2}\right|_{t_{n+1}} + O(\Delta t^2), \tag{30}$$

which shows that the leading error when using the exact mesh velocity at t_{n+1} instead of the D-GCL is only first order accurate with Δt.

Exercise 2.3 Write the Taylor series expansions up to degree 4 about $t_{n+1/2}$ for x^n and x^{n+1} and determine the leading error when using the exact velocity at $t_{n+1/2}$.
Do the numerical results obtained in Exercise 2.2 correspond to the analytical results?
Which error measure is more precise: ϵ_1 (determining the error only based on the final result) or ϵ_2 (determining the error based on all time steps during the simulation)?

Next we perform a similar analysis on the BDF2 scheme.

Exercise 2.4 Write the D-GCL you derived for the BDF2 scheme in terms of the face locations χ^{n-1}, χ^n, χ^{n+1} and the coefficients $\vec{\alpha}$.
Implement the D-GCL into `dgcl_BDF2.m`. Verify you implemented the D-GCL correctly so that the constant solution is preserved.
Perform and present a similar analysis as was done in Exercises 2.2, to determine graphically the order of consistency.

3 Simulation of uniform flow on a moving two-dimensional mesh

In this exercise, the D-GCL is applied to two-dimensional mesh motion. The mesh contains 9 cells (Fig. 4), and the motion that is prescribed is a rotation of the cell that is in the center of

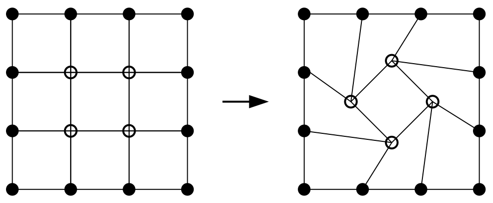

Figure 4: Two dimensional mesh.

the domain. Again we use (2) as governing equation and assume that the solution is a constant scalar u so that the flux $F(u)$ drops out of the equation. We therefore obtain

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_i} u \mathrm{d}V - \oint_{\partial\Omega_i}\left[u\frac{\mathrm{d}\vec{x}}{\mathrm{d}t}\right]\cdot\vec{n}\mathrm{d}S = 0. \tag{31}$$

The boundary integral now becomes a sommation over 4 cell faces (edges):

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_i} u \mathrm{d}V - \sum_{j=1}^{4}\left(u\frac{\mathrm{d}\vec{x}}{\mathrm{d}t}\cdot\vec{n}S\right)_j = 0, \tag{32}$$

and again we use a simple average to define the solution u at the internal faces

$$u_j = \frac{u_{cell1} + u_{cell2}}{2}, \tag{33}$$

as all of the internal faces have two neighboring cells. For the boundary faces we just use the value of the neigboring cell. Discretization in time with a Backward Euler scheme yields

$$\frac{u_i^{n+1}V_i^{n+1} - u_i^n V_i^n}{\Delta t} - \sum_{j=1}^{4}\left(u\frac{\mathrm{d}\vec{x}}{\mathrm{d}t}\cdot\vec{n}S\right)_j^{n+1} = 0, \tag{34}$$

which means that we need the mesh velocities $\frac{\mathrm{d}\vec{x}}{\mathrm{d}t}$ at the cell faces, which are determined from the DGCL. The system to be solved in (34) can also be written as

$$L\mathbf{u}^{n+1} = R\mathbf{u}^n. \tag{35}$$

Matlab code

The matlab code that is available to solve this problem can be found in `matlab\DGCL-2D\`and consists of the following files:

`dgcl_BE_2D.m` This is the main program; it takes the following steps:

- determine node locations,
- create cells by assigning 4 nodes for each cell,
- create connectivity information (for each cell face determine neighbor cells),

- intialization of data (solution, mesh velocities, cell volumes),
- for each time step:
  1. compute displacements of nodes,
  2. move the mesh (update node locations),
  3. compute new volumes,
  4. compute mesh velocity,
  5. construct L, R and solve system to obtain new solution.

`create_mesh.m` Creates the cell datastructure (each cell consists of 4 nodes)

`compute_cellvol.m` Computes the volume (in 2D surface) of the cell defined by 4 nodes.

`compute_connectivity.m` Determines the neighboring cells for all faces.

`compute_facecenter.m` Computes the center of the face (or edge).

`compute_faceSN.m` Computes the face normal and surface (in 2D this is the outward pointing normal and edge length).

`compute_meshVelocity.m` Computes the mesh velocity for the faces.

`movemesh.m` Updates the mesh nodes to the new location.

`plotMesh.m` Displays the mesh and creates colors depending on skewness of the cells.

`plotSolution.m` Displays the solution on the given mesh.

Mesh velocity computation

The mesh velocity computation is obtained from the displacement of a cell face from t_n to t_{n+1}, see Fig. 5. From (12) we know that the normal velocity component of the mesh velocity should

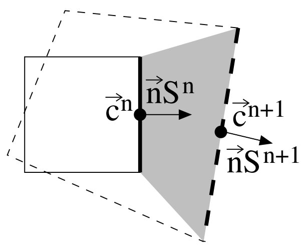

Figure 5: Two dimensional mesh.

satisfy

$$(\frac{\mathrm{d}\vec{x}}{\mathrm{d}t} \cdot \vec{n})^{n+1} = \frac{\Delta V_{face}^{n+1}}{\Delta t S^{n+1}},$$

therefore, a finite difference of the displacement of the face center

$$(\frac{\mathrm{d}\vec{x}}{\mathrm{d}t})^{n+1} = \frac{\vec{c}^{n+1} - \vec{c}^{n}}{\Delta t}, \tag{36}$$

does not generally satisfy the D-GCL.

Example Use the following settings: number of time steps per period `NdtP = 10` and number of periods to simulate `Np = 2`, and run the script `dgcl_BE_2D.m`. From the results you can observe that the solution does not remain uniform. Using smaller time steps by increasing `NdtP = 20, 40` shows that the error decreases approximately with $O(\Delta t)$.

Exercise 3.1 Write the swept volume ΔV_{face}^{n+1} in terms of the geometric parameters $\vec{n}S^{n+1}$, $\vec{n}S^{n}$, $\vec{c}^{n+1}$, $\vec{c}^{n}$ and write an expression for $(\frac{\mathrm{d}\vec{x}}{\mathrm{d}t})^{n+1}$ that satisfies the D-GCL.
The expression derived above can be implemented into `compute_meshVelocity.m`, verify that the solution remains uniform.

BONUS Exercise 3.2 (optional) *This exercise is optional: you can receive 1 point bonus.* Modify the Matlab code for integration with the ESDIRK-method. You can do this by making a loop over the number of ESDIRK stages within the loop for the time steps. The time level in a stage k is computed by $t_k = t_n + c_k\Delta t$, where $c_k = \sum_{i=1}^{k} a_{ki}$. The coefficients for a 4-stage ESDIRK scheme can be found in App. A. Write the swept volume $\Delta V_{face}^{(k)}$ in terms of the geometric parameters $\vec{n}S^{(k)}$, $\vec{n}S^{n}$, $\vec{c}^{(k)}$, $\vec{c}^{n}$ and write an expression for $(\frac{\mathrm{d}\vec{x}}{\mathrm{d}t})^{(k)}$ that satisfies the D-GCL.

A Butcher tableau for 4-stage ESDIRK scheme

Table 2: Third order ESDIRK

c_1	0	0	0	0
c_2	$\frac{1767732205903}{4055673282236}$	$\frac{1767732205903}{4055673282236}$	0	0
c_3	$\frac{2746238789719}{10658868560708}$	$-\frac{640167445237}{6845629431997}$	$\frac{1767732205903}{4055673282236}$	0
c_4	$\frac{1471266399579}{7840856788654}$	$-\frac{4482444167858}{7529755066697}$	$\frac{11266239266428}{11593286722821}$	$\frac{1767732205903}{4055673282236}$
b_i	$\frac{1471266399579}{7840856788654}$	$-\frac{4482444167858}{7529755066697}$	$\frac{11266239266428}{11593286722821}$	$\frac{1767732205903}{4055673282236}$