First Mandatory Assignment

MEK4420

Simon Lederhilger

January 28th 2025

Scope of the assignment

The assignment at hand is to solve the integral equation

$$\pi\phi(\boldsymbol{\varkappa}) = \int_{\partial\Omega} \left(\phi\partial_{\hat{\boldsymbol{n}}}\,\mathrm{G} - \mathrm{G}\,\partial_{\hat{\boldsymbol{n}}}\phi\right)\mathrm{d}S$$

by numerical implementation, calculate the added mass, and compare the results to theoretical values. The shapes to be considered are a circle, ellipses of aspect ratios $a/b = 1/2$ and $a/b = 1/10$, and a square. As we shall need to consider the theoretical values of these shapes' added masses, these will be derived. Although the lecture notes found in the repository in which this assignment is located already have an explanation of added mass, albeit rather heuristically inclined, a different approach will be taken for this assignment. This assignment will furthermore function as documentation for the `Python`-code developed for the purpose of calculating added mass.

Added mass on bodies

To gain insight into what *added mass* actually represents, the reader is highly encouraged to read the 1953 paper of Charles G. Darwin *Note on Hydrodynamics.*[1] It is therein demonstrated that bodies moving in an ideal fluid cause drift of the fluid, and that the drift volume is equivalent to the added mass. Authors in older works usually provide the kinetic energy of the system instead of the added mass, as the added mass may be inferred. Consider a body in an unbounded fluid being accelerated such that its energy is T_{body}. We know this induces a drift in the fluid, whose energy we shall label T_{fluid}, and that the total energy then is $T = T_{\text{body}} + T_{\text{fluid}}$. Consider now a fluid flow with potential Φ. Letting ϱ denote the fluid density, the kinetic energy of the fluid is then

$$T_{\text{fluid}} = \frac{\varrho}{2}\int_{\Omega} q^2\,\mathrm{d}V, \quad q = |\nabla\Phi| = \frac{\mathrm{d}w}{\mathrm{d}\mathfrak{z}}\frac{\mathrm{d}w^*}{\mathrm{d}\mathfrak{z}^*},$$

integrating in the entire fluid domain Ω, excluding the body. The complex formulation will be necessary later, but we pay no attention to it for the time being. The speed q may be related to the velocity magnitude U by assuming the modal superposition $\Phi(t;\boldsymbol{x}) = \boldsymbol{U}(t)\cdot\boldsymbol{\phi}(\boldsymbol{x})$. It is apparent that in general, the energy may be expressed as

$$T = \frac{(m+m')U^2}{2}, \qquad m' = \varrho k A,$$

where we note the added mass m' depends on the area of the body A, and an inertia coefficient k. This coefficient indicates the orientation of the body in relation to the direction of motion—we can imagine an ellipse moving parallel to its major axis will cause less drift in the fluid than it would moving perpendicular to it. That will at least that will be shown to be the case, mathematically. One may confirm the dependency on orientation readily in one's own kitchen—if imagination does not satisfy—by filling a container with water, and holding a spoon so that the bowl is submerged, moving it every which way. One shall find that moving the spoon with the bowl perpendicular to the direction of motion requires more work than moving it with the bowl parallel to the direction of motion. In fact, maintaining such work with a force F, we have that $FU = \partial_t T_{\text{fluid}}$, and extending Blasius' theorem,[2] we have that

$$\boldsymbol{F} = -\partial_t\boldsymbol{U} : \varrho\int_{\partial\Omega}\boldsymbol{\phi}\otimes\hat{\boldsymbol{n}}\,\mathrm{d}S,$$

where S is the contour of the body. $\boldsymbol{\phi}$ is a vector containing the modes of translation,

$$\boldsymbol{\phi}(\boldsymbol{x},t) = \phi_1\hat{\boldsymbol{\imath}} + \phi_2\hat{\boldsymbol{\jmath}} + \phi_6\hat{\boldsymbol{\omega}}_{\hat{\boldsymbol{k}}}.$$

This same formulation is found in the lecture notes, yielding the added mass tensor in terms of the following integral, which will be used for the numerical calculation.

$$\mathfrak{m} = \varrho\int_{\partial\Omega}\boldsymbol{\phi}\otimes\hat{\boldsymbol{n}}\,\mathrm{d}S$$

[1] [2] Darwin (1953)

[2] [5] Milne–Thomson, pp.255–256

Discrete integral equation

The integral equation

Since the fluid is ideal, we may reduce the incompressibility condition to the LAPLACE equation,

$$\nabla^2\phi = 0, \qquad \text{in } \Omega.$$

It may be shown with the product rule that for any harmonic functions $\phi, \psi \in \Omega$,

$$\nabla \cdot \big(\phi\nabla\psi - \psi\nabla\phi\big) \equiv 0,$$

which upon being integrated over Ω and applying GAUSS' divergence theorem, yields

$$\int_{\partial\Omega} \big(\phi\nabla\psi - \psi\nabla\phi\big)\,\mathrm{d}S = 0.$$

Since we only need the values of the potential on the contour $\partial\Omega$ to calculate the added mass, we look to GREEN functions, which are defined through the property that they satisfy some differential operator except at some point $\boldsymbol{\varkappa} \in \mathrm{Q}$, where $\mathrm{Q} = \Omega \cup \partial\Omega$, such that

$$\nabla^2\,\mathrm{G} = \delta(\boldsymbol{x} - \boldsymbol{\varkappa}), \qquad \boldsymbol{x} \in \mathrm{Q}.$$

In other words, the GREEN function is almost harmonic, and we expect the above integral should hold except at the pole. The GREEN function for the LAPLACE operator is the natural logarithm,

$$\mathrm{G}(\boldsymbol{x}) = \ln r, \qquad r = |\boldsymbol{x} - \boldsymbol{\varkappa}|,$$

which has a pole at $\boldsymbol{\varkappa}$. Now, by placing $\boldsymbol{\varkappa} \in \partial\Omega$, we find by the CAUCHY principal value,[1] that

$$-\pi\phi(\boldsymbol{\varkappa}) + \mathrm{pv}\int_{\partial\Omega} \phi\partial_{\hat{\boldsymbol{n}}}\ln r\,\mathrm{d}S = \mathrm{pv}\int_{\partial\Omega} \partial_{\hat{\boldsymbol{n}}}\phi\ln r\,\mathrm{d}S.$$

An outline of calculating the principal value is found in the lecture notes from January 21st. We drop the principal value notation for brevity.

The boundary element method

To implement the integral equation numerically, we utilize the boundary element method. In essence, we distribute a number of nodes along the boundary on which we would like to solve the integral equation, and linearily interpolate the points to approximate boundary. If the number of nodes is N, then we have $\partial\Omega \sim S = \{S_n \colon n \leq N,\, n \in \mathbb{Z}^+\}$. We then assume the potential is constant on each of S_n, equal to the potential evaluated at the midpoint, labelling this ${\phi_j}^n \equiv \phi_j(\boldsymbol{\varkappa}_n)$. Now, since the normal derivative of the potential also must be zero on the line segments, the integrals in the integral equation may be approximated by the sum of the integrals evaluated over wach of the line segments as follows.

$$\int_{\partial\Omega} \phi\partial_{\hat{\boldsymbol{n}}}\ln r\,\mathrm{d}S \approx \sum_{n=1}^{N} \phi^n \int_{S_n} \partial_{\hat{\boldsymbol{n}}}\ln r\,\mathrm{d}S \tag{1}$$

$$\int_{\partial\Omega} \ln r\partial_{\hat{\boldsymbol{n}}}\phi\,\mathrm{d}S \approx \sum_{n=1}^{N} \hat{\boldsymbol{n}}^n \int_{S_n} \ln r\,\mathrm{d}S \tag{2}$$

It should be quite clear that the integral equation may then be written as the matrix equation

$$-\pi\phi^n + \sum_{n=1}^{N} \phi^n \mathtt{\Theta}_{m,n} = \sum_{n=1}^{N} \hat{\boldsymbol{n}}^n \mathtt{h}_{m,n},$$

where we have set $\mathtt{\Theta}_{m,n}$ and $\mathtt{h}_{m,n}$ to be approximations of equations (1) and (2), respectively.

The logarithmic gradient

It turns out the integral of the gradient of the logarithm can be determined using complex analysis. The gradient is an operator from the real numbers into the vector space of the complex numbers, in the sense that

$$\nabla u(\mathfrak{z}) = \partial_x u(\mathfrak{z}) + i\partial_z u(\mathfrak{z}), \qquad \mathfrak{z} = x + iz.$$

Considering now an analytic function $\boldsymbol{u} = u + iv$, its complex derivative is given by $\partial_{\mathfrak{z}}\boldsymbol{u} = \partial_x u + i\partial_x v$, which by the CAUCHY–RIEMANN equations yields that $\partial_{\mathfrak{z}}\boldsymbol{u} = \nabla u^*$. We recall that the principal value of the complex logarithm is given by $\log(\mathfrak{z}) = \ln|\mathfrak{z}| + i\,\mathrm{Arg}(\mathfrak{z})$,[2] so that $\partial_{\mathfrak{z}}\log(\mathfrak{z}) = \nabla\ln|\mathfrak{z}|^*$.

Now, for a curve parametrized by $\lambda(s)$, the normal vector $\hat{\boldsymbol{n}}$ may be represented by[3]

$$\nu(s) = -i\lambda'(s) = \frac{\mathrm{d}z}{\mathrm{d}s} - i\frac{\mathrm{d}x}{\mathrm{d}s} = -i\frac{\mathrm{d}\mathfrak{z}}{\mathrm{d}s}.$$

The inner product may be defined such that $\boldsymbol{u}\cdot\hat{\boldsymbol{n}} = \mathrm{Re}\,(\boldsymbol{u}^*\nu(s))$. We then have that

$$\partial_{\hat{\boldsymbol{n}}}\ln r = \mathrm{Re}\left(-i\frac{\mathrm{d}\mathfrak{z}}{\mathrm{d}s}\partial_{\mathfrak{z}}\log(\mathfrak{z} - \boldsymbol{\varkappa}_m)\right).$$

Since the line segment is parameterized by s, the differential element may then be taken over $\mathfrak{z}$. By the linearity of the $\mathrm{Re}\,(\star)$ operator, we have that

$$\int_{S_n} \partial_{\hat{\boldsymbol{n}}}\ln r\,\mathrm{d}S = -\,\mathrm{Re}\,i\int_{\boldsymbol{x}_{n-1}}^{\boldsymbol{x}_n} \frac{\mathrm{d}}{\mathrm{d}\mathfrak{z}}\log(\mathfrak{z} - \boldsymbol{\varkappa}_m)\,\mathrm{d}\mathfrak{z}.$$
$$= \mathrm{Arg}\left(\frac{\boldsymbol{x}_{n-1} - \boldsymbol{\varkappa}_m}{\boldsymbol{x}_n - \boldsymbol{\varkappa}_m}\right).$$

This is just the angle between $\boldsymbol{x}_n$, $\boldsymbol{\varkappa}_m$, and $\boldsymbol{x}_{n-1}$.

[1] [3] LAVRENTEV & ŠABAT, pp.331–332
[2] [3] LAVRENTEV & ŠABAT, p.30
[3] [4] MARKUŠEVIČ, p.175

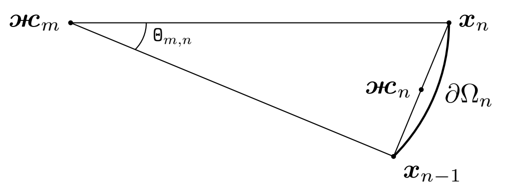

Figure 1: Visualization of $\Theta_{m,n}$. Here $\partial\Omega_n$ denotes the segment along $\partial\Omega$ between $\boldsymbol{x}_n$ and $\boldsymbol{x}_{n-1}$.

If $\boldsymbol{x}$ then is the array of the N nodes x_n, we can fill the matrix Θ with the arguments with `for`-loops, which is implemented in the `assemble` method in the `IntegralEquation` class, found in `integralequation.py`. The class variable $\boldsymbol{\varkappa} = 1/2(\boldsymbol{x}_n + \boldsymbol{x}_{n-1})$, containing the aforementioned midpoints is created upon initialization of the class, since multiple methods make use of it.

Algorithm 1 Assemble Θ

```
for $i, j \le N$ do
    if $i = j$ then
        $\Theta_{ij} \leftarrow -\pi$
    else
        $\Theta_{ij} \leftarrow \texttt{angle}\left(\frac{x[j-1]-\varkappa[i]}{x[j]-\varkappa[i]}\right)$
    end if
end for
```

This method essentially assembles the entire left-hand side of the integral equation, so that we may simply solve $\phi = \Theta^{-1}\mathtt{h}$.

Gauss–Legendre quadrature

To construct $\mathtt{h}$, we evaluate the integral of the logarithm using quadrature. For an integral on the real line, Gaussian quadrature approximates it by transforming it to the unit ball,

$$\int_a^b y(x)\,\mathrm{d}x \mapsto \int_{-1}^1 \eta(\xi)\,\mathrm{d}\xi,$$

where $\eta(\xi) \equiv y\big(x(\xi)\big)\partial_\xi x$, a change of variables such that $x(-1) = a$ and $x(1) = b$. Gauss–Legendre quadrature of order K has us approximating this integral by

$$\sum_{k=1}^K w_k \eta(\xi_k), \quad w_k = \frac{2}{\left(1-{\xi_k}^2\right)\left(\mathrm{P}'_K(\xi_k)\right)^2},$$

where P_K is the Kth Legendre polynomial, ξ_k is its kth zero, and w_k is the corresponding weight. The Legendre polynomials are given by the recursion formula[1]

$$(n+1)\,\mathrm{P}_{n+1}(x) = (2n+1)x\,\mathrm{P}_n(x) - n\,\mathrm{P}_{n-1}(x),$$

where we have defined $\mathrm{P}_0(x) = 1$, $\mathrm{P}_1(x) = x$. From the recurrence relation we find the 2nd order Legendre polynomial, its zeros, and its weights:

$$\mathrm{P}_2(x) = \frac{3x^2-1}{2}, \quad \xi_k = (-1)^k\frac{\sqrt{3}}{3}, \quad w_k = 1$$

We find below that we have no use for the 3rd polynomial, but we may use the 4th, for which we have

$$\mathrm{P}_4(x) = \frac{105x^4 - 90x^2 + 9}{24}$$

We may parametrize the variable of integration by writing that the line segment S_n is given by

$$\mathfrak{z}(\xi) = \left(\frac{\boldsymbol{x}_n - \boldsymbol{x}_{n-1}}{2}\right)\xi + \left(\frac{\boldsymbol{x}_n + \boldsymbol{x}_{n-1}}{2}\right),$$

where $\xi \in [-1,1]$. We recognize the rightmost term above as $\boldsymbol{\varkappa}_n$, and for brevity, we write that $\boldsymbol{x}_n - \boldsymbol{x}_{n-1} = \boldsymbol{\delta x}$, which is implemented as the method `IntegralEquation.Δx`. Now $\ln|\mathfrak{z}(\xi)|$ clearly is a function from the real numbers into the real numbers, so we may use the fact that for any real valued function over the complex plane,

$$\int_C f(\mathfrak{z})\,\mathrm{d}S = \int_a^b f\big(\mathfrak{z}(\xi)\big)|\mathfrak{z}'(\xi)|\,\mathrm{d}\xi.$$

We calculate that $|\mathfrak{z}'(\xi)| = 1/2|\boldsymbol{\delta x}|$, which is implemented as the class variable `dS` in the `Quadrature` class of `quadrature.py`.

Approximated added mass

Let $\delta S_n = |\boldsymbol{x}_n - \boldsymbol{x}_{n-1}|$.

$$\mathfrak{m} = [m_{ij}] \approx \varrho\sum_{n=1}^N {\phi_j}^n {\hat{n}_i}^n \delta S_n.$$

Added mass of a circle

Let a be the radius of the circle. We then have that its potential as a result of horizontal translation is given by

$$w = a^2U\mathfrak{z}^{-1} = \Phi + i\Psi, \quad \mathfrak{z} = x + iz.$$

We see that

$$\phi_1 = \mathrm{Re}\left(a^2U\mathfrak{z}^{-1}\right) = \frac{a^2U\cos\theta}{r^2};$$

$$\phi_2 = \mathrm{Re}\left(a^2Ui\mathfrak{z}^{-1}\right) = \frac{a^2U\sin\theta}{r^2}.$$

By calculating the gradient of ϕ and transforming into polar coordinates, we find that

$$q_r = \frac{a^2U\cos\theta}{r^2}, \quad q_\theta = \frac{a^2U\sin\theta}{r^2}, \quad q^2 = \frac{a^4U^2}{r^4}.$$

[1][1] Abramowitz & Stegun, 22.7.10, p.782

We may now calculate the kinetic energy of the fluid by integrating over the entire fluid domain Ω,

$$T_{\text{fluid}} = \frac{\varrho}{2}\int_0^{2\pi}\int_a^\infty q^2 r\,\mathrm{d}r\mathrm{d}\theta = \frac{\pi\varrho a^2 U^2}{2}.$$

Rotating the cylinder ought not induce drift in the fluid, as there is no mechanisms by which the fluid should be compelled to move from the circle turning. Because of the rotational symmetry of the circle, we must have that

$$\mathfrak{m} = [m_{ij}] = \begin{bmatrix} \pi\varrho a^2 & 0 & 0 \\ 0 & \pi\varrho a^2 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

For å diskretisere sirkelen konstruerer vi to `linspace`, nemlig $\theta_{\mathtt{p}}$ og $\theta_{\mathtt{m}}$, for å så lage to tupler med punkter som ligger på sirkelen. Vi definerer dem på følgende vis.

$$\theta_{\mathtt{p}} = \mathtt{linspace}\left(\frac{2\pi}{\mathtt{N}}, 2\pi, \mathtt{N}\right)$$

$$\theta_{\mathtt{m}} = \mathtt{linspace}\left(0, \frac{2\pi(\mathtt{N}-1)}{\mathtt{N}}, \mathtt{N}\right)$$

Her er `N` antall noder. Vi kan nå definere punkter på sirkelen ved å definere $\boldsymbol{x} = R_0(\cos\theta, \sin\theta)$, for både $\theta_{\mathtt{p}}$ og $\theta_{\mathtt{m}}$. Vi ser at $\theta_{\mathtt{m}}$ er etterslepende, fordi vi da kan finne midtpunktet ved å ta gjennomsnittet mellom dem. Det er i disse midtpunktene vi vil evaluere ϕ.

Det er hensiktsmessig å lage en klasse for å løse intergrallikningen, og midtpunktene bør være en metode her. Vi definerer dermed klassen `IntegralEquation` i filen `integralequation.py`. Vi har da metoden $\boldsymbol{\varkappa} \rightarrow$ `tuple`, som er definert ved $\boldsymbol{\varkappa} = 1/2(\boldsymbol{x}_{\mathtt{p}} + \boldsymbol{x}_{\mathtt{m}})$. For å sette sammen venstre side av integrallikningen, trenger vi en metode for å konstruere $\Theta \in \mathbb{R}^{\mathtt{N}\times\mathtt{N}}$. Vi gjør dette med `for`-løkker over indeksene `i` og `j`.
Grunnet maskinregningsfeil, kan `argument` ende opp med verdier litt høyere enn 1, si $1 + \varepsilon$, hvor vi typisk får at $\varepsilon = O(10^{-8})$. Som en følge, vil Numpy vise en feilmelding i `numpy.arccos`. Ved å legge inn en toleranse, unngår vi feilmeldingen, og koden kompilerer. Toleransen bør sjekkes slik at man både ser at avviket er mindre enn toleransen *og* at argumentet faktisk er større enn 1.

For å regne ut høyresiden av integrallikningen, må vi sette sammen matrisen `h`. Den inneholder en tilnærming av integralet til logaritmefunksjonen over sirkelbuen mellom $\boldsymbol{x}_{\mathtt{p}}$ og $\boldsymbol{x}_{\mathtt{m}}$. Denne tilnærmingen er gjort med en to-punkts Gausskvadraturmetode. Utledningen og referanser er å finnes i forelesningsnotatene. Denne matrisen skal skalarmultipliseres med moden vi er ute etters normalvektor. Vi lager en metode som finner normalvektoren. Siden alle normalvektorene ligger på todimensjonale linjestykker, vil normalvektoren være gitt ved $\hat{\mathtt{n}} = |(\Delta\boldsymbol{x}, \Delta\boldsymbol{y})|^{-1}(-\Delta\boldsymbol{y}, \Delta\boldsymbol{x})$. Vi lager en metode som setter sammen høyresiden av likningen, slik at løsningen kan finnes med `numpy.linalg.solve`.

I en ny fil kan vi konstruere koordinatene for en sirkel, og finne addert masse. I figur 2, ser vi at den teoretiske adderte massen i sjette mode er identisk oppfylt. Dette er grunnet definisjonen på normalvektoren på en sirkelgeometri—den samsvarer med posisjonsvektoren, så deres kryssprodukt blir null. Konvergensen til den adderte massen i første og andre mode er relativt god. Ser man på verdiene, virker det som en dobling i N vil føre til en reduksjon i relativ feil på faktor fire.

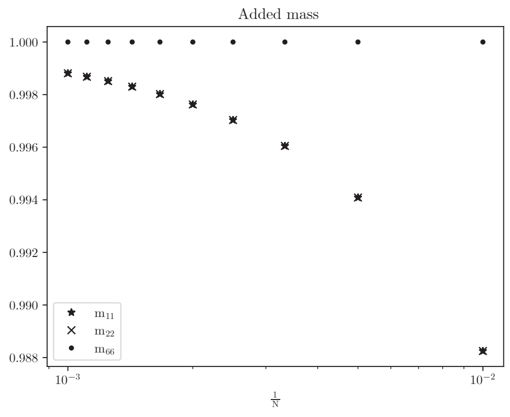

Figure 2: Numerisk utregnet over teoretisk addert masse på en sirkel.

Første modes potensial kan vi finne analytisk, som sett i første forelesningsnotat.

$$\phi_1 = -{R_0}^2\partial_x \ln r = -\cos\theta,$$

når vi er på randa til enhetssirkelen. Samsvaret mellom analytisk og numerisk løsning er utmerket for høye verdier av N, som vi ser i figur 3.

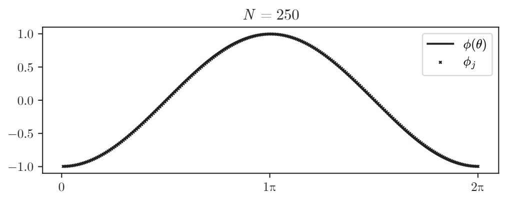

Figure 3: ϕ_1 er gitt analytisk med heltrukket linje, og numerisk med punkter.

Vi kan finne feilen i den numeriske løsningen ved å sjekke L^2-normen:

$$\left(\|\phi_j - \phi(\theta)\|_{L^2}\right)^2 = \int_\Omega \left(\phi_j - \phi(\theta)\right)^2 \mathrm{d}x.$$

Vi ser at den numeriske løsningen allerede blir veldig god når vi kun har 100 noder på sirkelen.

N	ϕ_1 feil	ϕ_2 feil
10	.547075	.501985
20	.285168	.272773
30	.192806	.187301
40	.145664	.142582
50	.117055	.115091
60	.097843	.096484
70	.084050	.083054
80	.073666	.072906
90	.065566	.064967
100	.059072	.058587

Andre modes potensial kan vi også finne analytisk på samme vis som første modes, dog med deriverte i y. Vi har altså at $\phi_2 = -\sin\theta$.

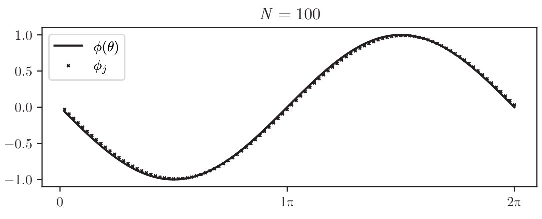

Figure 4: ϕ_2 er gitt analytisk med heltrukken linje, og numerisk med punkter.

Vi formoder at sjette modes potensial er identisk lik null—vi har ingen heft, så rotasjonen av en rotasjonssymmetrisk geometri bør ikke forflytte fluid. Den numeriske utregningen vår sier seg enig i dette, som sett i figure 5.

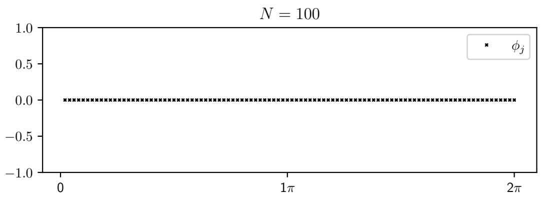

Figure 5: ϕ_6 er gitt numerisk med punkter.

Added mass of an ellipse

For å regne ut potensial og addert masse på en ellipse kan vi heller modifisere koden brukt til en sirkel, enn å skulle lage en helt ny kode. Vi sier nå at $\boldsymbol{x} = (a\cos\theta, b\sin\theta)$, hvor a er store halvakse, og b er lille halvakse. Den adderte massen konvergerer fint for både $a/b = 2$ og $a/b = 10$, som vi ser i figurer 6 og 7.

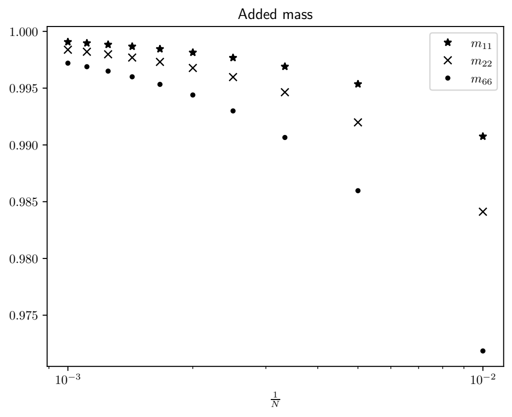

Figure 6: Addert masse for ellipse med $a/b = 2$.

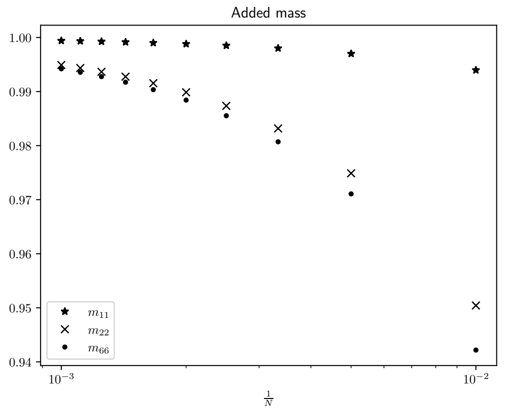

Figure 7: Addert masse for ellipse med $a/b = 10$.

Den dårligere konvergensen for større a kan være grunnet større mellomrom mellom nodene. Siden disse nodene faller nedenfor den faktiske ellipsebuen, vil normalvektoren heller ikke være helt riktig. Dessuten vil diskretiseringen til en sirkel alltid være en regulær flerkant, hvis normalvektorer nødvendigvis peker inn mot origo, som forklarer hvorfor konvergensen er dårligere enn sirkelen. Her ser vi ut til å ha den samme konvergensraten, med at en dobling i N tilsvarer en fjerdedel av relativ feil.

Vi blir ikke spurt om å sammenlikne det numeriske potensialet koden produserer med den analytiske, så vi går ut ifra at de gode resultatene vi så for sirkelen overføres til ellipsen.

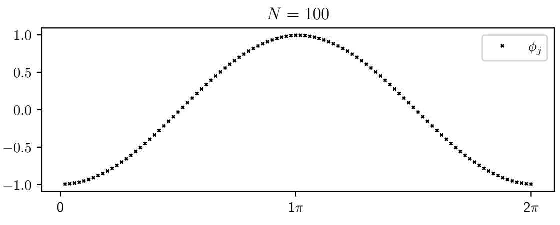

Figure 8: ϕ_1 for ellipse med $a/b = 2$.

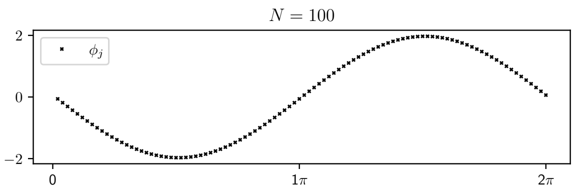

Figure 9: ϕ_2 for ellipse med $^a/_b = 2$.

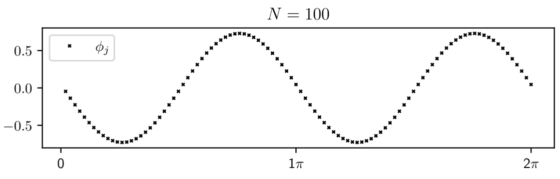

Figure 10: ϕ_6 for ellipse med $^a/_b = 2$.

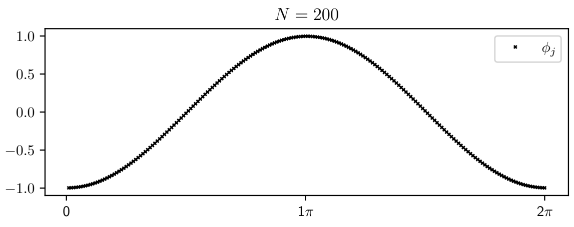

Figure 11: ϕ_1 for ellipse med $^a/_b = 10$.

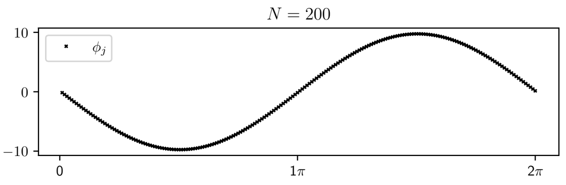

Figure 12: ϕ_2 for ellipse med $^a/_b = 10$.

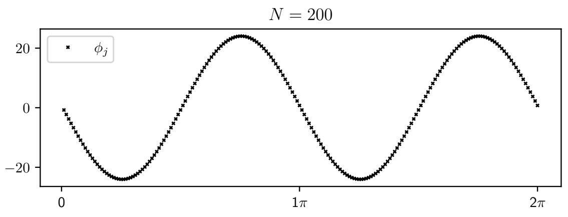

Figure 13: ϕ_6 for ellipse med $^a/_b = 10$.

Amplituden til første mode ser ut til å tilsvare b, og andre mode a.

Added mass of a square

Å skulle diskretisere kvadratet er ikke like trivielt som sirkelen eller ellipsen. Vi ønsker å skulle kunne gjenbruke samme klasse for integrallikningen, så en polar implementering av diskretiseringen er mest opplagt. En kontinuerlig mulighet er en såkalt superellipse, som kan diskretiseres med

$$\boldsymbol{x}(\theta) = \left(|\cos\theta|^{2/N} \operatorname{sgn}(\cos\theta), |\sin\theta|^{2/N} \operatorname{sgn}(\sin\theta) \right),$$

som vil konvergere raskt mot et kvadrat når N blir stor. Problemet her er at θ ikke faktisk er vinkelen i parametriseringen. Parametriseringsvariablen vil bruke lang tid i hjørnene på superellipsen, og en enorm andel av normalvektorene langs randa vil derfor være fullstendig feil.

En enklere løsning vil være å heller kjøre θ gjennom `if`-sjekker, som følger.

Algorithm 2 Konstruer kvadrat

```
for $n \le \texttt{N}$ do
    if $\theta_n \in [-\pi/4, \pi/4)$ then
        $\boldsymbol{x}_n, \boldsymbol{y}_n \leftarrow 2a, 2a\tan(\theta_n)$
    else if $\theta_n \in [\pi/4, 3\pi/4)$ then
        $\boldsymbol{x}_n, \boldsymbol{y}_n \leftarrow 2a\sec(\theta), 2a$
    else if $\theta_n \in [3\pi/4, 5\pi/4)$ then
        $\boldsymbol{x}_n, \boldsymbol{y}_n \leftarrow -2a, -2a\tan(\theta_n)$
    else if $\theta_n \in [5\pi/4, 7\pi/4)$ then
        $\boldsymbol{x}_n, \boldsymbol{y}_n \leftarrow -2a\sec(\theta), -2a$
    end if
end for
```

Denne løsningen er ikke veldig vakker, men den forsikrer at alle nodene faktisk ligger på kvadratet. Dette kan ikke sies for hjørnene i $\boldsymbol{\varkappa}$, ettersom hjørnenodene kan havne innenfor randa i dersom ikke enten $\boldsymbol{x}_p$ eller $\boldsymbol{x}_m$ ligger akkurat i hjørnet for denne N-verdien. Vi ser i figur 14 at dette kan føre til merkelige mønstre i konvergensen til den adderte massen, hvor tilnærmingen faktisk kan bli dårligere for en større N-verdi.

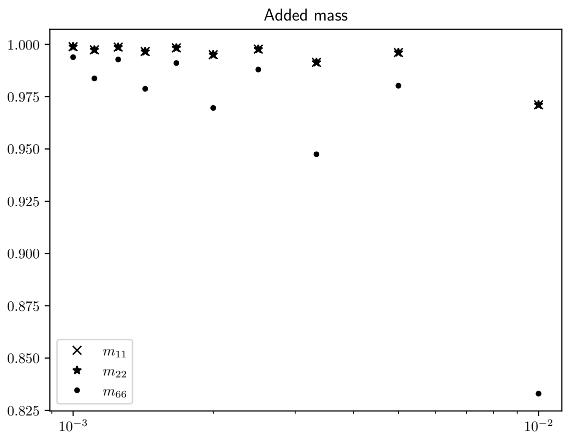

Figure 14: Numerisk utregnet over teoretisk addert masse på et kvadrat.

Samtidig ser vi at den utregnede adderte massen fakitsk konvergerer. Vil man unngå å se denne unøyaktigheten, kan man forsikre seg at $\{\pi/4, 3\pi/4, 5\pi/4, 7\pi/4\} \in \theta$ ved å la N alltid være delelig på 8, slik som i figur 15.

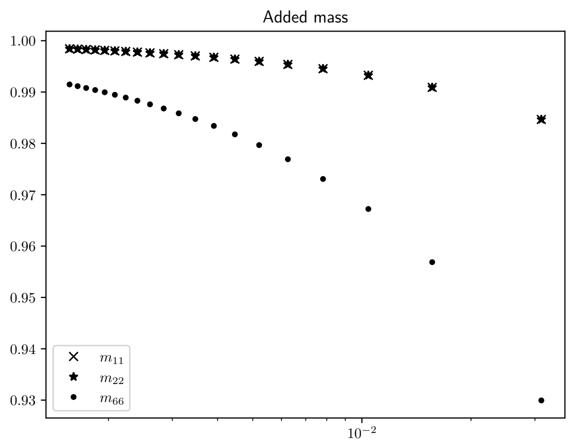

Figure 15: Numerisk utregnet over teoretisk addert masse på et kvadrat med N delelig på 8.

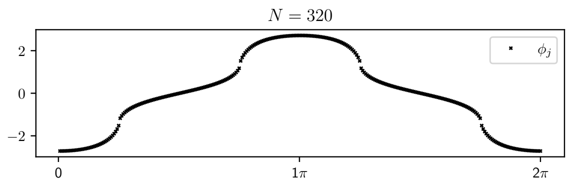

Figure 16: ϕ_1 er gitt numerisk med punkter.

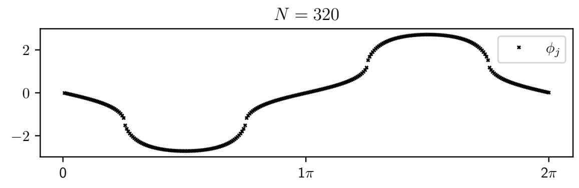

Figure 17: ϕ_2 er gitt numerisk med punkter.

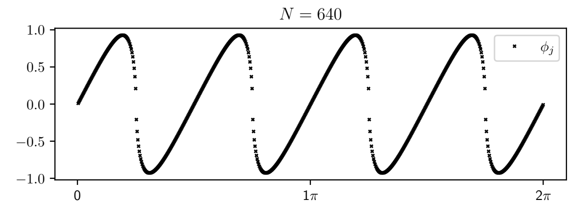

Figure 18: ϕ_6 er gitt numerisk med punkter.

References

[1] ABRAMOWITZ, Milton and STEGUN, Irene A. *Handbook of Mathematical Functions.* 3rd ed. National Bureau of Standards, 1965.

[2] DARWIN, Charles Galton. "Note on Hydrodynamics". In: *Proceedings of the Cambridge Philosophical Society* 49 (1953), pp. 342–354.

[3] LAVRENTEV, Mihail Alekseyevič (ЛАВРЕНТЬЕВ) and ŠABAT, Boris Vladimirovič (ШАБАТ). *Methoden der komplexen Funktionentheorie.* VEB Deutscher Verlag der Wissenschaften, 1967.

[4] MARKUŠEVIČ, Aleksey Ivanovič (МАРКУШЕВИЧ). *Theory of Functions of a Complex Variable, Volume II.* Ed. by Richard A. SILVERMAN. Prentice-Hall, inc., 1965.

[5] MILNE–THOMSON, Louis Melville. *Theoretical Hydrodynamics.* Macmillan & co. ltd., 1968.