**Programming C++**

Project **Numeric Integration**

Status: 13. Mai 2022, 17:36

Supervisor: Dr. E. Karabelas, elias.karabelas@uni-graz.at

# 1 Einleitung

Die Integration von Funktionen auf beliebigen Polyedern in $\mathbb{R}^d$ bildet ein Grundwerkzeug vieler Simulationssoftware (z.b.: Ansys FLUENT, StarCCM+, Abaqus,...). Exakte Integration ist selten effizient möglich weswegen auf *numerische Integration* zurückgegriffen wird. Das bekannteste numerische Verfahren zur Approximation von Integralen ist die *Gauss-Quadratur*.[1] Dieses Projekt soll Ihnen einen Einblick geben was bei kommerzieller Software *under the hood* vorgeht. Zum besseren Verständnis gibt es zu Beginn eine kurze Zusammenfassung der wichtigsten mathematischen Konzepte.

**Gauss-Quadratur:** Dies ist die gängiste Method um Integrale der Form $\int_{-1}^{1} f(x)dx$ zu approximieren. Die Idee ist simpel:

$$\int_{-1}^{1} f(x)\mathrm{d}x \approx \sum_{i=1}^{n} w_i f(x_i). \tag{1}$$

Das Integral wird durch eine gewichtete Summe von Funktionsauswertungen approximiert. Die natürliche Zahl $n$ nennt man auch Ordnung. Die Zahlen $w_i$ heißen *Integrationsgewichte* und die Punkte $x_i$ heißen *Stützstellen*. Nimmt man für $x_i$ einfach zufällige gleichverteilte Punkte in $[-1, 1]$ und $w_i = 1$ erhält man die *Monte-Carlo-Integration* welche in sehr hohen Dimensionen Sinn macht. Für die Gauss-Quadratur möchte man aber mit möglichst wenig Punkten auskommen. Ohne auf die Details einzugehen (siehe z.b. Wikipedia Link), lässt sich zeigen, dass für die optimale Wahl der $x_i$ man auf die Nullstellen der *Legendre-Polynome* $P_n(x)$ v[2]om Grade $n$. Ebenso lassen sich die optimalen Gewichte $w_i$ über die Ableitung der Legendre-Polynome als

$$w_i = \frac{2}{(1 - x_i^2)[P_n'(x_i)]^2}$$

angeben. Es gilt weiters, dass eine Gauss-Quadratur der Ordnung $n$ Polynome bis zum Grade $2n - 1$ exakt integriert, also für $f(x)$ gegebenes Polynom.

Für dieses Projekt stellen wir Routinen zur Verfügung welche die Stützstellen $x_i$ und die Gewichte $w_i$ für das Intervall $[0, 1]$ berechnen (siehe `tools.h`).

**Gauss-Jacobi-Quadratur:** Dies ist eine Verallgemeinerungd[3]er Gauss-Quadratur um Integrale der Form

$$\int_{-1}^{1} f(x)(1 - x)^{\alpha}(1 + x)^{\beta}\mathrm{d}x \tag{2}$$

[1] `https://en.wikipedia.org/wiki/Gaussian_quadrature`

[2] `https://en.wikipedia.org/wiki/Legendre_polynomials`

[3] `https://en.wikipedia.org/wiki/Gauss%E2%80%93Jacobi_quadrature`

mit $\alpha, \beta \in (-1, \infty)$ zu bestimmen (dies wird noch nützlich sein!). Hierbei ergeben sich die Stützstellen $x_i$ als Nullstellen der *Jacobi-Polynome*. Ebenso gilt für die Gauss-Jacobi-Quadratur der Ordnung $n$ das Polynome bis zum Grade $2n-1$ exakt integriert werden. Für dieses Projekt stellen wir Routinen zur Verfügung welche die Stützstellen $x_i$ und die Gewichte $w_i$ für das Intervall $[0,1]$ berechnen (siehe `tools.h`).

**Mehrdimensionale Substitutionsregel bei Integralen:** Oft ist es möglich numerische Integration für einfache Gebiete zu definieren (z.B.: auf einem Quadrat), jedoch möchte man aber auf anderen komplexeren Gebieten ein Integral bestimmen. Hier ist die Substitutionsregel in $\mathbb{R}^d$ mit $d \geq 2$ hilfreich: Sei $U \subset \mathbb{R}^d$ und $\phi\colon U \to \mathbb{R}^d$ eine *injektive* und stetig differenzierbare Funktion. Dann gilt:

$$\int_{\phi(U)} f(\mathbf{x})\mathrm{d}\mathbf{x} = \int_U f(\phi(\mathbf{u}))|\mathrm{det}J(\mathbf{u})|\mathrm{d}\mathbf{u}, \tag{3}$$

mit der Jakobi-Matrix

$$J := \begin{pmatrix} \frac{\partial \phi_1}{\partial x_1} & \cdots & \frac{\partial \phi_1}{\partial x_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial \phi_d}{\partial x_1} & \cdots & \frac{\partial \phi_d}{\partial x_d} \end{pmatrix}. \tag{4}$$

Damit lassen sich Integrale über Gebiete die als Bild eines einfachen Gebietes dargestellt werden können einfach berechnen. Ein Beispiel wird im nächsten Abschnitt erklärt.

**Duffy-Transformation:** Dies ist eine komfortable Möglichkeit Integrale über ein Dreieck als Integral über ein Quadrat darzustellen. Sei das sogenannte *Referenz-Dreieck* $\blacktriangle$ in $\mathbb{R}^2$ gegeben durch die drei Punkte $(0,0), (1,0), (0,1)$. Dann definiert man die *Duffy-Transformation* als

$$\begin{aligned} &\phi\colon [0,1]^2 \to \blacktriangle, \\ &(x_1, x_2) \mapsto (x_1, x_2(1-x_1)). \end{aligned}$$

Die Jacobi-Matrix (4) lässt sich ermitteln als

$$J = \begin{pmatrix} 1 & 0 \\ -x_2 & 1-x_1 \end{pmatrix}.$$

Damit gilt $\det(J) = 1 - x_1 \geq 0$ für $x_1 \in [0,1]$ und es folgt, dass für ein Integral über $\blacktriangle$

$$\int_{\blacktriangle} f(y_1, y_2)\mathrm{d}y_1\mathrm{d}y_2 = \int_{x_1=0}^{1}\int_{x_2=0}^{1} f(x_1, x_2(1-x_1))(1-x_1)\mathrm{d}x_1\mathrm{d}x_2$$

gilt. Wir hoffen Sie haben hier schon einen Zusammenhang mit der Gauss-Jacobi-Quadratur hergestellt ($(1-x_1) = (1-x_1)^{\alpha=1}$).

## 2 Projektaufgaben

Alle Aufgaben sollten mit den Datentypen `double` und `float` funktionieren, es wird also am besten sein gleich mit Templates und einem Template-parameter `T` zu arbeiten!

1. Numerische Integration auf einem beliebigen rechteckigen Gebiet $[a,b] \times [c,d] \subset \mathbb{R}^2$: (3 Pkt.)
   Ausgehend von den Routinen in `tools.h` sollen Sie eine Klasse `NumericIntegrator2D` konstruieren, welche Ihnen für eine gegebene Funktion $f(x_1, x_2)$ das Integral über ein Gebiet $[a,b] \times [c,d]$ bestimmt. Diese Klasse soll folgende Routinen realisieren:

   - `generate_integration_points_weights`: Eine Routine welche als Input eine Ordnung als `size_t` entgegennimmt und Integrationspunkte sowie Gewichte auf $[0,1]^2$ erstellt. Hierfür wird die vorhandene Routine `computeGaussRule` nützlich sein. *Hinweis:* in 1D hat man eine Summe in der Gauss-Quadratur, in 2D wird das zu einer Doppelsumme $\sum_{i=1}^{n}\sum_{j=1}^{n}\ldots$. Überlegen Sie zuerst wie das aussehen muss und erst dann wie man hier die Punkte und Gewichte aus der 1D Quadratur generieren kann. Als Vorstellungshilfe, machen Sie sich eine Zeichnung des Gebietes $[0,1]^2$ und markieren Sie für die $x$-Achse und $y$-Achse jeweils die Gausspunkte. Danach betrachten Sie das Integral

     $$\int_0^1 \int_0^1 f(x_1, x_2) \mathrm{d}x_1 \mathrm{d}x_2$$

     jeweils für einen fixen Gauss-Punkt $x_1^i$ und $x_2^j$ gesondert und schreiben auf wie die 1D-Quadratur für $f(x_1^i, x_2)$ und $f(x_1, x_2^j)$ aussieht. Danach sollten Sie relativ schnell erkennen wie Sie vorgehen müssen. Sollten es Ihnen hiermit nicht möglich sein die Punkte zu konstruieren können Sie den Projektleiter kontaktieren und um die Lösung dieses Unterproblems bitten zum Kostenpunkt von (-2Pkt).
   - `calculate_integral`: Diese Routine soll eine funktion $f(x_1, x_2)$ in der Form `std::function<T(T,T)>`, das Rechteckgebiet $[a,b] \times [c,d]$ als `std::array< std::pair<T,T> , 2>`, und die Integrationsordnung als `size_t` entgegen nehmen und das Integral bestimmen. Sie haben in 1.) schon die Integrationspunkte und Gewichte für den Spezialfall $[a,b] \times [c,d] = [0,1]^2$ bestimmt. Jetzt müssen Sie sich überlegen wie man hier die mehrdimensionale Substitution für Integrale möglichst geschickt einsetzt. *Hinweis:* Überlegen Sie sich Abbildungen welche das Interval $[0,1]$ auf $[a,b]$ und $[c,d]$ abbilden und konstruieren Sie daraus $\phi$. Damit bekommen Sie die Jakobi-Matrix und sollten erkennen was zu tun ist.

   Testen Sie Ihre numerische Integration auf beliebigen Rechtecken mit verschiedenen Ordnungen. Folgende Tests sollten auf jeden Fall durchgeführt werden:

   - $[a,b] \times [c,d] = [\frac{1}{4}, \frac{1}{2}] \times [\frac{1}{2}, \frac{3}{2}]$ und $f(x_1, x_2) = x_1^7 x_2^5 + x_1 x_2$. Verifizieren Sie hier, dass ab einer gewissen Ordnung (welche sollte das sein?) der numerischen Integration der exakte Wert von $\frac{794167}{8388608} \approx 0.0946721$ erreicht wird.
   - $[a,b] \times [c,d] = [-1,1]^2$ und $f(x_1, x_2) = \sin(\pi x_1)\sin(\pi x_2)$. Der exakte Wert des Integrals ist 0. Kann dieser erreicht werden, wenn nein warum?

2. Numerische Integration auf beliebigen Dreiecken: Erweitern Sie Ihre Klasse (3 Pkt.)

`NumericIntegrator2D` um eine Integrationsroutine auf einem beliebigen nicht-degenierten Dreieck welches von 3 Punkten $p_0, p_1, p_2 \in \mathbb{R}^2$ aufgespannt wird. Dazu brauchen Sie auch eine Abbildung vom Referenz-Dreieck ▲ auf ein beliebiges Dreieck welche durch

$$\psi(x_1, x_2) := \begin{pmatrix} p_1^1 x_1 + p_2^1 x_2 + p_0^1(1 - x_1 - x_2) \\ p_1^2 x_1 + p_2^2 x_2 + p_0^2(1 - x_1 - x_2) \end{pmatrix} \tag{5}$$

gegeben ist. Hierbei ist $^1$ die jeweilige $x$-Koordinate und $^2$ die $y$-Koordinate. Implementieren Sie hierzu eine Hilfsklasse `Triangle` welche folgende Funktionalität hat:

- Member `const std::array<std::array<T, 2>, 3> coords` welche die Koordinaten des Dreiecks hält
- Member `det` welche die Determinante der Jacobi-Matrix speichert. (Ausrechnen!)
- `public` Funktion `map_point` welche einen Punkt im Referenz-Dreieck auf einen Punkt im Dreieck abbildet.
- `public` Funktion `get_det` welche die Determinante der Jakobi-Matrix retourniert.
- Geeignete Konstruktoren und Destruktoren.

Damit können Sie nun ähnlich wie in 1.) Ihrer Klasse `Numeric2DIntegrator` um folgende Routinen erweitern:

- `generate_integration_points_weights_trig`: Ähnlich wie für das Referenz-Rechteck $[0, 1]^2$ müssen Sie eine geeignet Sammlung von Gauss-Punkten für das Referenz-Dreieck ▲ konstruieren. Hierfür werden Ihnen die Routinen `computeGaussRule` und `computeGaussJacobiRule` nützlich sein. *Hinweis:* Verwenden Sie zuerst die *Duffy-Transformation* als Abbildung des Einheitsquadrats auf das Referenz-Dreieck und gehen Sie analog wie in 1.) vor[4].
- `calculate_integral_trig`: Diese Routine soll eine funktion $f(x_1, x_2)$ in der Form `std::function<T(T,T)>`, das Dreieck $p_0, p_1, p_2$ als Klasse `Triangle`, und die Integrationsordnung als `size_t` entgegen nehmen und das Integral bestimmen. *Hinweis:* Mehrfache Anwendung des Subsitutionsprinzips wird vonnöten sein.

Testen Sie Ihre Integrationsroutine zuerst für das Referenz-Dreieck ▲ und die Funktion $f(x_1, x_2) = 1 - x_1 - x_2$. Das Integral sollte $\frac{1}{6}$ ergeben. Danach probieren Sie Ihre Routinen für die Funktionen aus 1.) aus.

3. **(Bonus) Integration auf Hexaedern:** Erweitern Sie Ihr Projekt um numerische Integration auf beliebigen Hexaedern $[a, b] \times [c, d] \times [e, f] \subset \mathbb{R}^3$. Hierfür können Sie komplett analog wie für das Rechteck vorgehen, Sie müssen nur eine Dimension mehr beachten. Hierfür wäre eine neue Integrationsklasse `Numeric3DIntegrator` zu entwickeln welche analog zu 1.) aufgebaut ist. Unterschied ist nur, dass wir uns im $\mathbb{R}^3$ aufhalten, also die Funktion für `calculate_integral` vom Typ `std::function<T(T,T,T)>`, und die Grenzen als `std::array< std::pair<T,T> , 3>` einzugeben sind. Als Test-Funktionen können Sie $f(x_1, x_2, x_3) := x_1 x_2 x_3$ verwenden auf dem Hexaeder $[\frac{1}{8}, 1] \times [\frac{1}{8}, \frac{1}{3}] \times [\frac{1}{4}, \frac{1}{2}]$. Das Ergebnis sollte $\frac{1155}{524288}$ sein. (+ 2 Pkt.)

[4] Lösungshinweis: Achtung der Integrand ändert sich durch die Duffy-Transformation, dies sollten Sie sich auf Papier überlegen warum das Gewicht in einer Axen-Richtung mit dem Faktor $(1 - x_1^j)$ multipliziert werden muss.

4. **(Superbonus) Integration auf Tetraedern:** Erweitern Sie Ihr Projekt um numerische Integration von beliebigen Tetraedern welche von 4 Punkten $(p_0, p_1, p_2, p_3) \in \mathbb{R}^3$ aufgespannt wird. Der *Referenz-Tetraeder* ist durch die 4 Punkte $(0,0,0), (1,0,0), (0,1,0), (0,0,1)$ gegeben. Hierfür werden Sie auch die dreidimensionale *Duffy-Transformation* benötigen (+ 3 Pkt.)

$$\begin{aligned} \phi\colon [0,1]^3 &\to \text{RefTet}, \\ (x_1, x_2, x_3) &\mapsto (x_1, x_2(1-x_1), x_3(1-x_1)(1-x_2)). \end{aligned}$$

Die Jacobi-Matrix (4) lässt sich ermitteln als

$$J = \begin{pmatrix} 1 & 0 & 0 \\ -x_2 & 1-x_1 & 0 \\ -(1-x_2)x_3 & -(1-x_1)x_3 & (1-x_1)(1-x_2) \end{pmatrix}.$$

Damit gilt $\det(J) = (1-x_1)^2(1-x_2) \geq 0$ für $x_1, x_2 \in [0,1]$ und es folgt, dass für ein Integral über das Referenztetraeder

$$\begin{aligned} &\int_{\text{RefTet}} f(y_1, y_2, y_3) \mathrm{d}y_1 \mathrm{d}y_2 \mathrm{d}y_3 \\ &\quad = \int_{x_1=0}^{1} \int_{x_2=0}^{1} \int_{x_3=0}^{1} f(x_1, x_2(1-x_1), x_3(1-x_1)(1-x_2))(1-x_1)^2(1-x_2) \mathrm{d}x_1 \mathrm{d}x_2 \mathrm{d}x_3 \end{aligned}$$

gilt. Das restliche Vorgehen ist analog zu 2.), Sie müssen eine Hilfsklasse `Tetrahedron` implementieren mit den gleichen Routinen wie `Triangle`. Die Abbildungsvorschrift um einen Punkt im Referenz-Tetraeder auf einen Punkt im gegebenen Tetraeder abzubilden lautet:

$$\psi(x_1, x_2, x_3) := \begin{pmatrix} p_1^1 x_1 + p_2^1 x_2 + p_3^1 x_3 + p_0^1(1 - x_1 - x_2 - x_3) \\ p_1^2 x_1 + p_2^2 x_2 + p_3^2 x_3 + p_0^2(1 - x_1 - x_2 - x_3) \\ p_1^3 x_1 + p_2^3 x_2 + p_3^3 x_3 + p_0^3(1 - x_1 - x_2 - x_3) \end{pmatrix} \tag{6}$$

Als Test können Sie die Funktion $f(x_1, x_2, x_3) = 1 - x_1 - x_2 - x_3$ nehmen, welche über das Referenztetraeder integriert $\frac{1}{24}$ ergibt.