



# Praktikum zu LV 7328 - Ruby:

## Übung 08

Ruby-Extensions in "C"



## Organisatorisches



- Arbeitsverzeichnis:  
`~/kurse/ruby/08/`
- Dateinamen:  
`08-xmath.rb` # Verfassen und abgeben  
`08-xmath.c` # Verfassen und abgeben  
`08-xmath.i` # Verfassen und abgeben
- Werkzeuge:  
`ruby` # Der Interpreter  
`emacs` # mit Ruby-Mode. NICHT X-



- Die Aufgabe besteht aus drei Teilen:
  - A: Nutzung vorhandener Bibliotheken**  
Erzeugen eines einfachen "C" Extension-Moduls, dabei Erschließung vorhandener "C"-Funktionen für Ruby.
  - B: Nutzung eigener C-Implementierungen**  
Erweiterung des Moduls um eine einfache selbstgeschriebene Funktion.
  - C: Erschließung von Performance-Vorteilen**  
Vergleichstest zur Rechenzeiten:  
Implementierung der Simpson-Formel zur numerischen Integration sowohl in Ruby als auch in "C",  
Vergleich des Laufzeitverhaltens beider Methoden.



- Mathematischer Hintergrund
  1. Eingebaute Funktionen:  
Einige der in der C-Bibliothek für mathematische Funktionen hat Ruby nicht per Modul "Math" bereitgestellt:

```
double cbrt(double); // Kubikwurzel
double log1p(double); // log(1+x)
```
  2. Abgeleitete Funktion:  
Die Gamma-Funktion ist eine Verallgemeinerung der Fakultätsberechnung. Sie steht auch auf C-Seite nicht direkt zur Verfügung, wohl aber ihr Logarithmus:

```
double lgamma(double); // log(Γ(x))
```
  3. Numerische Integration - die Simpson-Formel (mit  $\Delta x := (x_1 - x_0)/2n$ ):
$$\int_{x_0}^{x_1} f(x) dx \approx \frac{\Delta x}{3} [f(x_1) - f(x_0) + 2 \sum_{i=0}^{n-1} (f(x_0 + 2i \cdot \Delta x) + 2f(x_0 + (2i+2) \Delta x))] ]$$



## Die Aufgabe



### A, Allgemeine Beschreibung:

- Erzeugen Sie ein Extension-Modul namens "XMath"
- Es soll in Ruby folgende Funktionen zur Verfügung stellen  
`XMath::cbrt, XMath::loglp`
- Schreiben Sie ein Ruby-Programm (Datei 08-xmath.rb), das "XMath" verwendet, um für  $x = 0, 1, 2, \dots, 9$  eine Wertetabelle der beiden neu gewonnenen Funktionen auszugeben. Der Kern Ihrer Ausgabe kann so aussehen:

```
printf("%g\t\t", x)
printf("%.8f\t", XMath::cbrt(x))
printf("%.8f\n", XMath::loglp(x))
```

### B (\*), Allgemeine Beschreibung:

- Fügen Sie nun eine eigene Funktion "my\_gamma" hinzu und geben Sie auch deren Werte in der o.g. Tabelle aus (neue rechte Spalte).
- Die Funktion ist in "C" zu implementieren und von den vorhandenen Funktionen "lgamma" und "exp" abzuleiten.



## Die Aufgabe



### C, Allgemeine Beschreibung:

- Fügen Sie nun eine weitere Funktion "exp\_int" hinzu. Prototyp:  
`double exp_int(double x0, double x1, double n2);`
- Die Funktion soll die Simpson-Formel zur Integration der Funktion `exp` in den Grenzen `[x0, x1]` implementieren. Sei  $n2 = 2 * n$ .
- Implementieren Sie ferner das Simpson-Verfahren auch in Ruby.
- Ermitteln Sie den Integralwert zwischen  $x0=0$  und  $x1=1$  mit beiden Verfahren für  $n2 = 10, 1000, 100000$  und  $1000000$ . Stimmen die Werte der Verfahren überein? Wie groß sind die Differenzen zur exakten Lösung?
- Wie lange benötigt die Ruby-Methode im Vergleich zur C-Methode? Implementieren Sie einen "benchmark"-Report.



## Die Aufgabe



### A: Hinweise zum Vorgehen

- Legen Sie eine SWIG-Quelldatei Datei "08-xmath.i" an, die die notwendigen C-Prototypen der gewünschten Funktionen enthält, außerdem die explizite Anweisung `#include <math.h>`. Beachten Sie die Vorlesungsbeispiele, achten Sie insb. auf "**extern**" (!).

- Rufen Sie dann SWIG auf:

```
$ swig -ruby 08-xmath.i
```

Es sollte eine Datei `08-xmath_wrap.c` resultieren.

- Erzeugen Sie eine Datei "Makefile" mit Hilfe von Ruby, wie in der Vorlesung vorgestellt, und führen Sie "make" aus. Sie sollten eine Datei "XMath.so" erhalten.

- Starten Sie nun Ihr Ruby-Programm; die Anweisung

```
require "XMath"
```

sollte nun funktionieren, und Sie sollten eine Wertetabelle erhalten.



## Die Aufgabe



### B: Hinweise zum Vorgehen

- Implementieren Sie die Funktion

```
double my_gamma(double x)
```

in Datei "08-xmath.c" im aktuellen Arbeitsverzeichnis.

- Nehmen Sie den neuen Prototypen auf in `08-xmath.i` und lassen Sie SWIG erneut laufen.

- WICHTIG: Erzeugen Sie ein neues Makefile!

- Rufen Sie erneut "make" auf --> neue Datei `XMath.so` !

- Erweitern Sie in `08-xmath.rb` nun die Wertetabelle um die neue Spalte für `my_gamma(x)`.



## C: Hinweise zum Vorgehen

- Implementieren Sie nun auch die C-Funktion  

```
double exp_int(double x0, double x1, int n2)
```

in Datei "08-xmath.c" im aktuellen Arbeitsverzeichnis.
- Nehmen Sie auch diesen Prototypen auf in 08-xmath.i. Lassen Sie SWIG und "make" erneut laufen.
  
- Implementieren Sie die analoge Top-Level Ruby-Methode "exp\_int".
- Auswertung: Ermitteln Sie in Ruby  

```
int1 = exp_int( 0.0, 1.0, n )  
int2 = XMath::exp_int( 0.0, 1.0, n)
```

jeweils für die angegebenen Werte von n, geben Sie die Ergebnisse aus, ebenso die Abweichungen vom exakten Ergebnis (e-1).
  
- Zeitmessungen: Verwenden Sie einfach das Modul "benchmark", vgl. Pickaxe-Buch S.210ff (Regex-Abschnitt)