Implementación del método IRK basada en la vectorización SIMD

IRKGL16-simd

M. Antoñana, J. Makazaga and A. Murua

eman ta zabal zazu
Universidad del País Vasco Euskal Herriko Unibertsitatea

juliacon 2022

July 27th-29th

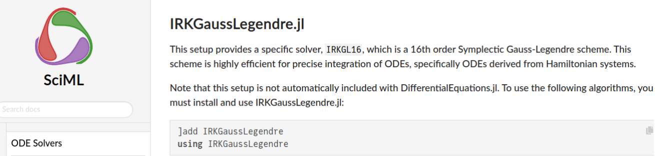
Contenidos

1. Quienes somos
2. IRKGL-simd
3. Benchmarks
4. Conclusiones y trabajo futuro

Quienes somos

Área de investigación

Se enmarca dentro de la matemática aplicada y computacional, más concretamente, en el análisis e implementación de métodos avanzados para la integración numérica de problemas modelados por EDOs



SciML

Search docs

ODE Solvers

IRKGaussLegendre.jl

This setup provides a specific solver, `IRKGL16`, which is a 16th order Symplectic Gauss-Legendre scheme. This scheme is highly efficient for precise integration of ODEs, specifically ODEs derived from Hamiltonian systems.

Note that this setup is not automatically included with DifferentialEquations.jl. To use the following algorithms, you must install and use IRKGaussLegendre.jl:

```
]add IRKGaussLegendre
using IRKGaussLegendre
```

Instituciones

eman ta zabal zazu

Universidad del País Vasco | Euskal Herriko Unibertsitatea

FACULTY OF COMPUTER SCIENCE
UNIVERSITY OF THE BASQUE COUNTRY

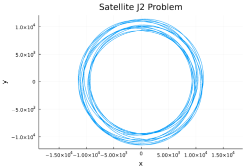
bcam
basque center for applied mathematics

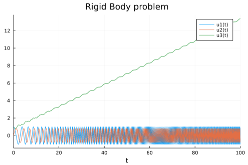
IRKGL16-simd

Idea central:
El método IRKGL16 es adecuado para aprovechar la vectorización incluida en los sistemas de computación modernos

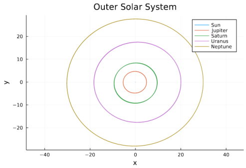
- **Objetivo**: mostrar que los métodos **RK implícitos** pueden ser más eficientes que los **RK explícitos** de DifferentialEquations.jl:
  - **Vern9**: EDOs generales de primer orden
  - **DPRKN12**: EDOs de 2º orden
- **Preliminar**: **paso de tamaño fijo** (futuro: adaptativo)
- **Problemas**: solución de problemas **no-rígidos en alta precisión** (tolerancias como $< 1e-10$)

EDOs generales de primer orden





EDOs de segundo orden



IRKGL16-simd

¿Qué siginifica IRKGL16?

- **Método de integración**: dado un problema de valor inicial de sistemas EDOs de la forma,

$$\frac{du}{dt} = f(t, u), \quad u(t_0) = u_0 \in \mathbb{R}^d$$

aplicamos un método de integración para obtener la **aproximación de la solución** $u_k \approx u(t_k)$,

$$u_{k+1} = IRKGL16(t_k, u_k, h_k) \quad at \quad t_{k+1} = t_k + h_k \quad for \quad k = 0, 1, 2, \ldots$$

- **Runge-Kutta** familia de métodos de **un solo paso**
- **Implícito**: para EDOs no rígidas, las ecuaciones implícitas se pueden resolver mediante **iteración de punto fijo** (implementación sencilla)
- **Gauss-Legendre**: los coeficientes que determinan el método, se obtienen aplicando la cuadratura de Gauss-Legendre (simplectico y simétrico)
- **Método de orden alto**: $s = 8$ etapas $\Rightarrow$ orden 16!!

IRKGL16-simd

Scientists must know about hardware to write fast code

¿Qué significa IRKGL16-simd?

El método IRKGL16 puede aprovechar la **paralelización** de los computadores modernos de dos formas:

- **Multi-tarea**: las etapas $s = 8$ en las fórmulas RK pueden evaluarse en paralelo (lo exploramos en el trabajo anterior)
- **Vectorización(SIMD)**: los cómputos que actúan sobre vectores con s=8 números Float64, pueden ser evaluados simultáneamente por CPUs modernas con registros especializados de 512 bits

$$\begin{bmatrix} Z_1 \\ Z_2 \\ Z_3 \\ Z_4 \\ Z_5 \\ Z_6 \\ Z_7 \\ Z_8 \end{bmatrix} = \sin\left(\begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \\ X_8 \end{bmatrix}\right) + 4 * \left(\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \\ Y_5 \\ Y_6 \\ Y_7 \\ Y_8 \end{bmatrix}\right)^{2/3}$$

Single Instruction Multiple Data
$\Leftarrow$ mismo costo que la versión escalar !!

IRKGL16-simd

SIMD.jl package: nos permite vectorizar explícitamente el código de IRKGL16 de forma sencilla

$$\text{Ejemplo:} \quad f(Y_i, p, t_i + hc), \quad i = 1, \ldots s$$

Una evaluación

```
nbody = 5
s = 8
W = rand(s,3,nbody,2)
Gm = rand(nbody)
ddW = similar(W)

q = W[1,:,:,:]
ddq = ddW[1,:,:,:]
@btime NbodyODE!(ddq, q, Gm, 0.)

> 79.291 ns (0 allocations: 0 bytes)
```

Ocho evalualiaciones vectorizadas

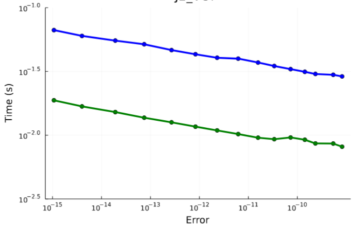
```
Q=VecArray{s,Float64,4}(W)
ddQ=VecArray{s,Float64,4}(ddW)
@btime NbodyODE!(ddQ, Q, Gm, 0.)

>179.826 ns (0 allocations: 0 bytes)
```

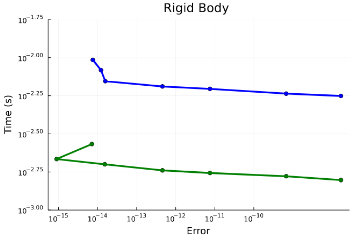
- Mejora de rendimiento: $79.291 * 8/179.826 \approx 3.5$
- Transparente para el usuario: misma implementación de la EDO

IRKGL16-simd

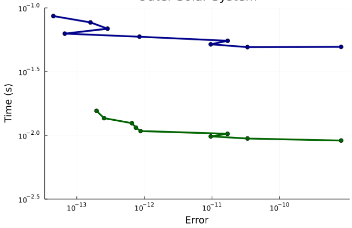
Benchmarks(I): IRKGL16 secuencial vs simd

EDOs de primer orden





EDOs de segundo orden



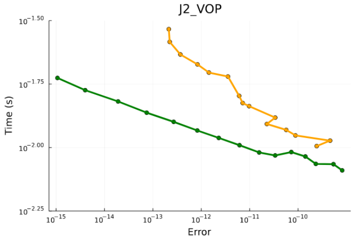
Perfomance improvement $\approx \times 3.5$

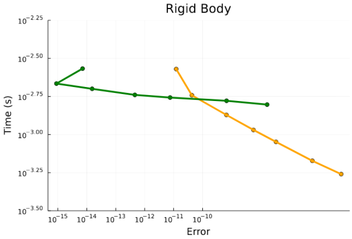
IRKGL16_seq
IRKGL16_simd

Benchmarks

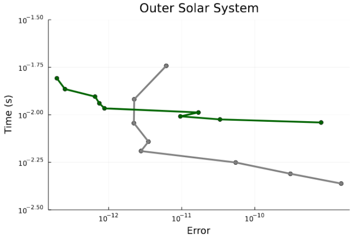
Comparativa: IRKGL16-simd vs Vern9//DPRKN12

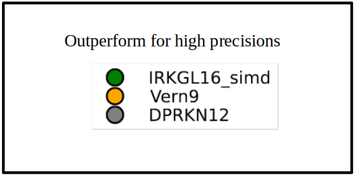
EDOs de primer orden





EDOs de segundo orden





Conclusiones y trabajo futuro

- **Conclusiones**
  - **El paquete SIMD.jl** nos permite explícitamente vectorizar el código IRKGL16 de **forma satisfactoria**
  - **IRKGL16-simd** mejora la eficiencia de los métodos RK explícitos de alto orden de DifferentialEquaitions.jl para computaciones en doble precisión y para tolerancias altas
  - **La vectorización SIMD** debe explorarse en otras aplicaciones
- **Trabajo futuro**
  - Añadir la opción de **paso de tamaño adaptativo-simétrico**
  - **Incorporaremos** al paquete IRKGaussLegendre.jl para cálculos de doble precisión.
  - **Simpléctico y simétrico**. Útil para aplicaciones de **Machine Learning** , ya que los gradientes se pueden calcular exactamente integrando las ecuaciones adjuntas hacia atrás

Useful References

- **DifferentialEquations.jl – A Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia**
  https://doi.org/10.5334/jors.151
- **Julia implementation of an implicit Runge-Kutta integrator IRKGL16**
  https://github.com/SciML/IRKGaussLegendre.jl
- **Explicit SIMD vectorization in Julia**
  https://github.com/eschnett/SIMD.jl
- **Single Instruction, Multiple Data (SIMD) in Julia**
  http://kristofferc.github.io/post/intrinsics/

Gracias!

y os animamos a utilizar nuestra implementación

- **Codigo preliminar**:
  `https://github.com/mikelehu/IRKGL_SIMD.jl`

- **Agradecimientos**

  - A los organizadores del congreso Juliacon2022
  - This work has reveived funding by the Spanish State Research Agency through project **PID2019-104927GB-C22 (GN-QUAMC)** and also from Department of Education of the Basque Government through **MATHMODE Research Group** (IT2494-19)

- **Contacto**: mikel.antonana@ehu.eus