SIMD-vectorized implementation of high order IRK integrators

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

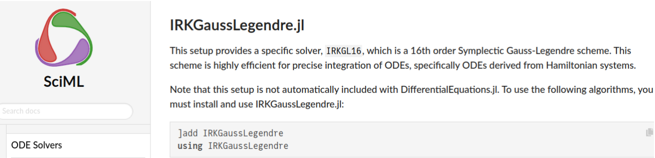
Outline

1. Who we are
2. IRKGL-simd
3. Benchmarks
4. Conclusions and future work

Who we are

Research area

We have a long experience of research in ***applied and computational mathematics****, with special focus on analysis and implementation of advanced methods for the numerical integration of problems modeled by ODEs.*


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
```

Institutions

eman ta zabal zazu

Universidad del País Vasco | Euskal Herriko Unibertsitatea

FACULTY OF COMPUTER SCIENCE

UNIVERSITY OF THE BASQUE COUNTRY

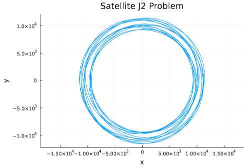
(bcam) basque center for applied mathematics

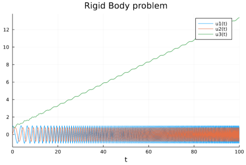
IRKGL16-simd

The central idea: IRKGL16 solver is well suited to take advantage of hardware vectorization

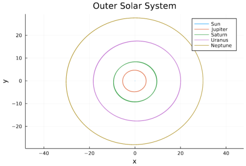
- **Goal**: show **implicit RK methods** can be more efficient than explicit recommended methods in DifferentialEquations.jl suite:
    - **Vern9**: for general Fist Order ODE
    - **DPRKN12**: for 2nd Order ODE
- **Preliminary**: **fixed step size** implementation (next adaptive)
- **Problems**: we focus on solving **non-stiff ODEs** with **high accuracy** (tolerances $< 1e - 10$)

General First Order ODE





Second Order ODE



IRKGL16-simd

What does mean IRKGL16?

- **Integration method**: giving an initial value problem of systems ODEs of the form,

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

  we apply a integration method to get the numerical approximation of the solution $u_k \approx u(t_k)$,

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

- **Runge-Kutta** methods belong to the class of one-step integrators for numerical solution of ODEs
- **Implicit**: for nonstiff ODEs implicit equations can be solved by fixed-point iteration (easy implementation)
- **Gauss-Legendre**: based on the Gauss-Legendre quadrature formula (symplectic and time-symmetry)
- **High order method**: $s = 8$ stages $\Rightarrow$ 16 order

IRKGL16-simd

Scientists must know about hardware to write fast code

What does mean IRKGL16-simd?

IRKGL16 solver can take advantage of modern computer technology:

- **Multi-threading based parallelism**: all $s = 8$ stages in the RK formulas can be evaluated in parallel (we explored that in a previous work)
- **SIMD-based parallelism**: computations acting on vectors with $s = 8$ Float64 numbers, can be evaluated simultaneously by modern CPUs with 512-bit specialized registers

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

Single Instruction Multiple Data
$\Leftarrow$ same cost as scalar version !!

IRKGL16-simd

SIMD.jl package: allows us to explicitly SIMD-vectorize IRKGL16 code

Example: $f(Y_i, p, t_i + hc), \quad i = 1, \ldots s$

One evaluation

```
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)
```

Eight vectorized evaluations

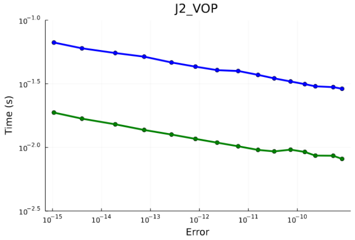
```
Q=VecArray{s,Float64,4}(W)
ddQ=VecArray{s,Float64,4}(ddW)
@btime NbodyODE!(ddQ, Q, Gm, 0.)

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

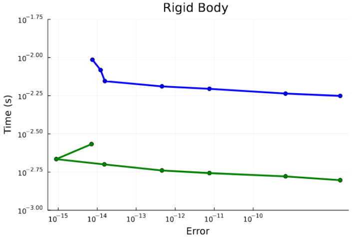
- Perfomance improvement: $79.291 * 8/179.826 \approx 3.5$
- Transparent for the user: same ODE implementation

IRKGL16-simd

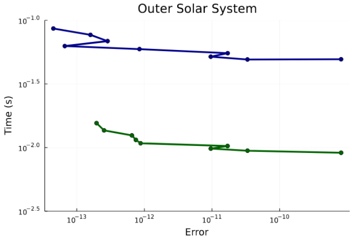
Benchmarks(I): IRKGL16 sequential vs simd

General First Order ODE





Second Order ODE



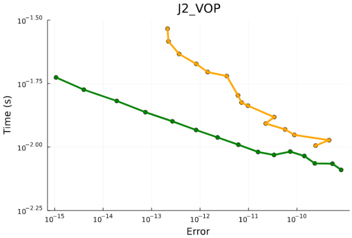
Perfomance improvement $\approx \times 3.5$

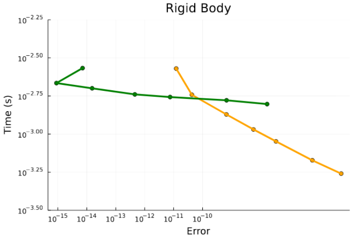
IRKGL16_seq
IRKGL16_simd

Benchmarks

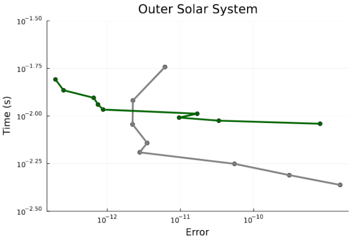
Benchmark: IRKGL16-simd vs Vern9//DPRKN12

General First Order ODE

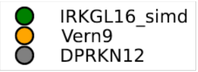




Second Order ODE



Outperform for high precisions



Conclusions and future work

- **Conclusions**
  - **SIMD.jl** package allows us to explicity SIMD-vectorize IRKGL16 code
  - **IRKGL16-simd** outperform high order explicit RK methods of DifferentialEquaitions.jl in double precision floating point for precision like $< 1e - 10$
  - **SIMD-vectorization** should be explored in other applications
- **Future work**
  - Apply **symmetric adaptive step** size strategy
  - **Add** IRKGL-simd implementation to IRKGaussLegendre.jl package for double precision computations
  - **Symplecticness and time-symmetry**. Useful for Scientific Machine Learning Applications: gradients can be exactly calculated by integrating backward in time the adjoint equations

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/

Thank you!

and we encourage you to use our implementation

- **Preliminary Code**:
  `https://github.com/mikelehu/IRKGL_SIMD.jl`

- **Acknowledgments**:
  - To the JuliaCon2022 organizers
  - 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)

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