Parallel & Distributed Computing: Lecture 4

Alberto Paoluzzi

October 11, 2018

1. Sequential implementation of domain integration of polynomials

Sequential implementation of domain integration of polynomials

Good contest exercize for “Loop Unrolling”

Loop transformation technique that attempts to optimize a program’s execution speed at the expense of its binary size, which is an approach known as space–time tradeoff.

The transformation can be undertaken manually by the programmer or by an optimizing compiler.

(from Wikipedia)

Normal loop	After loop unrolling

`for i := 1:8 do`
`    if i mod 2 = 0 then do_even_stuff(i)`
`                   else do_odd_stuff(i);`
`    next i;`
	`do_odd_stuff(1); do_even_stuff(2);`
	`do_odd_stuff(3); do_even_stuff(4);`
	`do_odd_stuff(5); do_even_stuff(6);`
	`do_odd_stuff(7); do_even_stuff(8);`

Figure 1: Loop unrolling

Domain integration of polynomials

Finite formulae for evaluation of integrals:

$$II_S \equiv \int\int_S f(\mathbf{p})\, dS, \qquad III_P \equiv \int\int\int_P f(\mathbf{p})\, dV, \tag{1}$$

The integrating function is a trivariate polynomial

$$f(\mathbf{p}) = \sum_{\alpha=0}^{n} \sum_{\beta=0}^{m} \sum_{\gamma=0}^{p} a_{\alpha\beta\gamma} x^\alpha y^\beta z^\gamma,$$

where α, β, γ are non-negative integers. Since the extension to $f(\mathbf{p})$ is straightforward, we focus on integrals of monomials:

$$II_S^{\alpha\beta\gamma} \equiv \int\int_S x^\alpha y^\beta z^\gamma\, dS, \qquad III_P^{\alpha\beta\gamma} \equiv \int\int\int_P x^\alpha y^\beta z^\gamma\, dV. \tag{2}$$

From Cattani, Paoluzzi. "Boundary integration over linear polyhedra", CAD, 1990

Basic integration functions

structure product over a polyhedral surface S, open or closed, is a summation of structure products (3) over the 2-simplices of a triangulation K_2 of S:

$$II_S^{\alpha\beta\gamma} = \int\int_S x^\alpha y^\beta z^\gamma \, dS = \sum_{\tau \in K_2} II_\tau^{\alpha\beta\gamma}$$

```
function II(P, alpha, beta, gamma, signedInt=false)
    w = 0
    V, FV = P
    if typeof(P) == PyCall.PyObject
        if typeof(V) == Array{Any,2}
            V = V'
        end
        if typeof(FV) == Array{Any,2}
            FV = [FV[k,:] for k=1:size(FV,1)]
            FV = FV+1
        end
    end
    if typeof(FV) == Array{Int64,2}
        FV = [FV[:,k] for k=1:size(FV,2)]
    end
    for i=1:length(FV)
        tau = hcat([V[:,v] for v in FV[i]]...)
        if size(tau,2) == 3
            term = TT(tau, alpha, beta, gamma, signedInt)
            if signedInt
                w += term
            else
                w += abs(term)
            end
        elseif size(tau,2) > 3
            println("ERROR: FV[$(i)] is not a triangle")
        else
            println("ERROR: FV[$(i)] is degenerate")
        end
    end
    return w
```

Basic integration functions

$$III_P^{\alpha\beta\gamma} = \int\int\int_P x^\alpha y^\beta z^\gamma \, dx\, dy\, dz$$

$$= \frac{1}{\alpha+1} \sum_{\tau \in K_2} \left[\frac{(\mathbf{a} \times \mathbf{b})_x}{|\mathbf{a} \times \mathbf{b}|} \right]_\tau II_\tau^{\alpha+1,\beta,\gamma}$$

```
function III(P, alpha, beta, gamma)
    w = 0
    V, FV = P
    if typeof(P) == PyCall.PyObject
        if typeof(V) == Array{Any,2}
            V = V'
        end
        if typeof(FV) == Array{Any,2}
            FV = [FV[k,:] for k=1:size(FV,1)]
            FV = FV+1
        end
    end
    for i=1:length(FV)
        tau = hcat([V[:,v] for v in FV[i]]...)
        vo,va,vb = tau[:,1],tau[:,2],tau[:,3]
        a = va - vo
        b = vb - vo
        c = cross(a,b)
        w += c[1]/vecnorm(c) * TT(tau, alpha+1, beta, gamma)
    end
    return w/(alpha + 1)
end
```

see cvdlab/LinearAlgebraicRepresentation.jl

Basic integration functions

$$
\begin{aligned}
II_\tau^{\alpha\beta\gamma} \quad = \quad & |\mathbf{a}\times\mathbf{b}| \sum_{h=0}^{\alpha} \binom{\alpha}{h} x_o^{\alpha-h} \cdot \\
& \cdot \sum_{k=0}^{\beta} \binom{\beta}{k} y_o^{\beta-k} \cdot \\
& \cdot \sum_{m=0}^{\gamma} \binom{\gamma}{m} z_o^{\gamma-m} \cdot \\
& \cdot \sum_{i=0}^{h} \binom{h}{i} a_x^{h-i} b_x^i \cdot \\
& \cdot \sum_{j=0}^{k} \binom{k}{j} a_y^{k-j} b_y^j \cdot \\
& \cdot \sum_{l=0}^{m} \binom{m}{l} a_z^{m-l} b_z^l \cdot \\
& \cdot II^{\mu\nu}
\end{aligned}
$$

```
function TT(tau::Array{Float64,2}, alpha, beta, gamma, signedInt=false)
  vo,va,vb = tau[:,1],tau[:,2],tau[:,3]
  a = va - vo
  b = vb - vo
  s1 = 0.0
  for h=0:alpha
    for k=0:beta
      for m=0:gamma
        s2 = 0.0
        for i=0:h
          s3 = 0.0
          for j=0:k
            s4 = 0.0
            for l=0:m
              s4 += binomial(m,l) * a[3]^(m-l) * b[3]^l *
                M( h+k+m-i-j-l, i+j+l )
            end
            s3 += binomial(k,j) * a[2]^(k-j) * b[2]^j * s4
          end
          s2 += binomial(h,i) * a[1]^(h-i) * b[1]^i * s3;
        end
        s1 += binomial(alpha,h) * binomial(beta,k) * binomial(gamma,m) *
            vo[1]^(alpha-h) * vo[2]^(beta-k) * vo[3]^(gamma-m) * s2
      end
    end
  end
  c = cross(a,b)
  if signedInt == true
    return s1 * vecnorm(c) * sign(c[3])
  else return s1 * vecnorm(c) end
end
```

Basic integration functions

$$II^{\alpha\beta} = \frac{1}{\alpha+1}\sum_{h=0}^{\alpha+1}\binom{\alpha+1}{h}\frac{(-1)^h}{h+\beta+1},$$

```
function M(alpha, beta)
    a = 0
    for l=1:(alpha + 2)
        a += binomial(alpha+1,l) * (-1)^l/(l+beta+1)
    end
    return a/(alpha + 1)
end
```