Recursive box and vertex integrations for one-loop hexagon reductions

Elise de Doncker[1] Junpei Fujimoto[2] Nobuyuki Hamaguchi [4] Tadashi Ishikawa[2] Yoshimasa Kurihara[2] Yoshimitsu Shimizu[3] Fukuko Yuasa[2]

[1]Department of Computer Science, Western Michigan University, Kalamazoo MI 49008, U. S.

[2]High Energy Accelerator Research Organization (KEK), Oho 1-1, Tsukuba, Ibaraki, 305-0801, Japan

[3]Graduate University for Advanced Studies, Hayama, Miura-gun, Kanagawa, 240-0193, Japan

[4]Hitachi, Ltd., Software Division, Totsuka-ku, Yokohama, Japan

tu-logo

ur-logo

Outline

1. Computational background
   - Iterated integration
   - Extrapolation
2. Hexagon reduction
   - Reduction n-dimensional N-point function
   - Triangle functions
   - Box functions

Outline

1. Computational background
   - Iterated integration
   - Extrapolation
2. Hexagon reduction
   - Reduction n-dimensional N-point function
   - Triangle functions
   - Box functions

Outline

tu-logo

ur-logo

Iterated integration

Integration over a product region $\mathcal{D} = \mathcal{D}_1 \times \ldots \times \mathcal{D}_\ell$,

Integral

$$If = \int_{\mathcal{D}_1} d\vec{x}^{(1)} \ldots \int_{\mathcal{D}_\ell} d\vec{x}^{(\ell)} \, f(\vec{x}^{(1)}, \ldots, \vec{x}^{(\ell)}),$$

implemented recursively using lower-dimensional code across successive groups of dimensions, $j = 1, \ldots, \ell$.

E.g., standard 1D integration code (such as DQAGE *from Quadpack[5]) can be used for 1D levels; or a combination of 1D and multivariate methods (such as* DCUHRE *[1]) across levels.*

Adaptive recursion levels

Algorithm at each recursion level

Evaluate initial region & update results
Initialize priority queue to empty
while (evaluation limit not reached
and estimated error too large)
Retrieve region from priority queue
Split region
Evaluate subregions & update results
Insert subregions into priority queue

Example

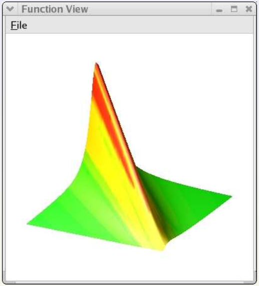

Figure:

$$\int_0^1 dx \int_0^1 dy \frac{2\alpha y}{(x+y-1)^2+\alpha^2} = \int_0^1 dx \left[\int_0^1 dy \frac{2\alpha y}{(x+y-1)^2+\alpha^2} \right]$$

Adaptive recursion levels

Algorithm at each recursion level

Evaluate initial region & update results
Initialize priority queue to empty
while (evaluation limit not reached
and estimated error too large)
Retrieve region from priority queue
Split region
Evaluate subregions & update results
Insert subregions into priority queue

Example

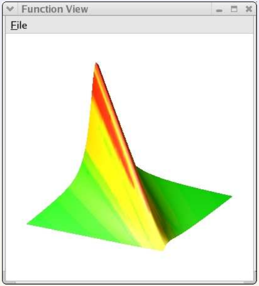

Figure:
$$\int_0^1 dx \int_0^1 dy \frac{2\alpha y}{(x+y-1)^2+\alpha^2} = \int_0^1 dx \left[\int_0^1 dy \frac{2\alpha y}{(x+y-1)^2+\alpha^2}\right]$$

Recursive vs. standard multivariate integration

$$\int_0^1 dx \int_0^1 dy \, \frac{2\alpha y}{(x+y-1)^2+\alpha^2}, \quad \alpha = 10^{-p}$$

	DQAGE × DQAGE		DCUHRE	
p	Abs. err.	# Eval.	Abs. err.	# Eval.
1	0.00e+00	21255	2.06e-12	144165
2	2.40e-13	93135	5.96e-12	1998675
3	3.49e-13	208035	1.37e-12	21040551
4	1.58e-13	388125	8.04e-12	99999963
5	4.49e-13	561585	4.40e-07	99999963
6	1.69e-09	527205	3.38e-02	99999963
7	1.42e-10	686745	1.99e+00	99999963
8	3.94e-10	902145	3.04e+00	99999963
9	3.20e-08	106965	3.13e+00	99999963
10	4.32e-09	1964385	3.14e+00	99999963
11	1.87e-01	58651365	3.14e+00	99999963

Recursive vs. standard multivariate integration

$$\int_0^1 dx \int_0^1 dy \frac{2\alpha y}{(x+y-1)^2+\alpha^2}, \quad \alpha = 10^{-p}$$

	DQAGE × DQAGE		DCUHRE	
p	ABS. ERR.	# EVAL.	ABS. ERR.	# EVAL.
1	0.00e+00	21255	2.06e-12	144165
2	2.40e-13	93135	5.96e-12	1998675
3	3.49e-13	208035	1.37e-12	21040551
4	1.58e-13	388125	8.04e-12	99999963
5	4.49e-13	561585	4.40e-07	99999963
6	1.69e-09	527205	3.38e-02	99999963
7	1.42e-10	686745	1.99e+00	99999963
8	3.94e-10	902145	3.04e+00	99999963
9	3.20e-08	106965	3.13e+00	99999963
10	4.32e-09	1964385	3.14e+00	99999963
11	1.87e-01	58651365	3.14e+00	99999963

Remarks

- The 2D integrand function has a ridge of height $2(1-x)/\alpha$ along $y=1-x$, which causes an increasingly difficult anomaly as the ridge becomes higher and steeper (with increasing p).
- The performance of DCUHRE deteriorates rapidly for $p>4$, with respect to accuracy and the number of subdivisions needed.
- With respect to memory use, for the maximum allowed number of function evaluations of maxpts = 100 million, the maximum number of regions that can be generated by successive bisections is 2,380,952 (using the integration rule of degree 7 with 21 points per region), and the work space for storing and managing the region collection needs to be for at least 19,047,634 doubles.

Remarks

- The 2D integrand function has a ridge of height $2(1-x)/\alpha$ along $y = 1 - x$, which causes an increasingly difficult anomaly as the ridge becomes higher and steeper (with increasing p).
- The performance of DCUHRE deteriorates rapidly for $p > 4$, with respect to accuracy and the number of subdivisions needed.
- With respect to memory use, for the maximum allowed number of function evaluations of maxpts = 100 million, the maximum number of regions that can be generated by successive bisections is 2,380,952 (using the integration rule of degree 7 with 21 points per region), and the work space for storing and managing the region collection needs to be for at least 19,047,634 doubles.

Remarks

- The 2D integrand function has a ridge of height $2(1-x)/\alpha$ along $y = 1 - x$, which causes an increasingly difficult anomaly as the ridge becomes higher and steeper (with increasing p).
- The performance of DCUHRE deteriorates rapidly for $p > 4$, with respect to accuracy and the number of subdivisions needed.
- With respect to memory use, for the maximum allowed number of function evaluations of maxpts = 100 million, the maximum number of regions that can be generated by successive bisections is 2,380,952 (using the integration rule of degree 7 with 21 points per region), and the work space for storing and managing the region collection needs to be for at least 19,047,634 doubles.

Remarks

- In comparison, the corresponding number of intervals for each direction of the 1D$\times$1D recursive integration with DQAGE$\times$DQAGE, allowing 10^4 evaluations in each coordinate direction (for 10^8 in 2D), is about 333 intervals (using the 15-point Gauss-Kronrod rules). This requires the space of about 1,500 doubles in 1D (for 2D, at most 3,000 doubles need to be in memory at any one time).

Outline

Extrapolation

- The integrals need to be obtained in the limit as a parameter $\to 0$.
- Methods for extrapolation to the limit $\mathcal{S}$ of a sequence $S(\varepsilon)$, as $\varepsilon \to 0$, rely on the existence of an asymptotic expansion

$$S(\varepsilon) \sim \mathcal{S} + a_1\varphi_1(\varepsilon) + a_2\varphi_2(\varepsilon) + \ldots$$

- Given a sequence $\{S(\varepsilon_\ell)\}$, an extrapolation is performed to create sequences that convergence faster than the original sequence.

tu-logo

ur-logo

Extrapolation

- The integrals need to be obtained in the limit as a parameter $\to 0$.
- Methods for extrapolation to the limit $\mathcal{S}$ of a sequence $S(\varepsilon)$, as $\varepsilon \to 0$, rely on the existence of an asymptotic expansion

$$S(\varepsilon) \sim \mathcal{S} + a_1\varphi_1(\varepsilon) + a_2\varphi_2(\varepsilon) + \ldots$$

- Given a sequence $\{S(\varepsilon_\ell)\}$, an extrapolation is performed to create sequences that convergence faster than the original sequence.

tu-logo

ur-logo

Extrapolation

- The integrals need to be obtained in the limit as a parameter $\to 0$.
- Methods for extrapolation to the limit $\mathcal{S}$ of a sequence $S(\varepsilon)$, as $\varepsilon \to 0$, rely on the existence of an asymptotic expansion

$$S(\varepsilon) \sim \mathcal{S} + a_1\varphi_1(\varepsilon) + a_2\varphi_2(\varepsilon) + \ldots$$

- Given a sequence $\{S(\varepsilon_\ell)\}$, an extrapolation is performed to create sequences that convergence faster than the original sequence.

Linear extrapolation

- A linear extrapolation method solves (implicitly or explicitly [3]) linear systems of the form

$$S(\varepsilon_\ell) = c_0 + c_1\varphi_1(\varepsilon_\ell) + \ldots c_\nu\varphi_\nu(\varepsilon_\ell), \quad \ell = 0, \ldots, \nu;$$

- i.e., systems of order $(\nu + 1) \times (\nu + 1)$ in unknowns $c_0, \ldots, c_\nu$ are solved for increasing values of ν.
- The coefficients $\varphi_k(\varepsilon_\ell)$ need to be known explicitly in order to apply this method.
- The computation for $\varphi_k(\varepsilon) = \varepsilon^k$ can be carried out recursively using Richardson extrapolation [4].

Linear extrapolation

- A linear extrapolation method solves (implicitly or explicitly [3]) linear systems of the form

$$S(\varepsilon_\ell) = c_0 + c_1\varphi_1(\varepsilon_\ell) + \ldots c_\nu\varphi_\nu(\varepsilon_\ell), \quad \ell = 0, \ldots, \nu;$$

- i.e., systems of order $(\nu + 1) \times (\nu + 1)$ in unknowns $c_0, \ldots, c_\nu$ are solved for increasing values of ν.
- The coefficients $\varphi_k(\varepsilon_\ell)$ need to be known explicitly in order to apply this method.
- The computation for $\varphi_k(\varepsilon) = \varepsilon^k$ can be carried out recursively using Richardson extrapolation [4].

tu-logo

ur-logo

Linear extrapolation

- A linear extrapolation method solves (implicitly or explicitly [3]) linear systems of the form

$$S(\varepsilon_\ell) = c_0 + c_1\varphi_1(\varepsilon_\ell) + \ldots c_\nu\varphi_\nu(\varepsilon_\ell), \quad \ell = 0, \ldots, \nu;$$

- i.e., systems of order $(\nu + 1) \times (\nu + 1)$ in unknowns $c_0, \ldots, c_\nu$ are solved for increasing values of ν.
- The coefficients $\varphi_k(\varepsilon_\ell)$ need to be known explicitly in order to apply this method.
- The computation for $\varphi_k(\varepsilon) = \varepsilon^k$ can be carried out recursively using Richardson extrapolation [4].

Linear extrapolation

- A linear extrapolation method solves (implicitly or explicitly [3]) linear systems of the form

$$S(\varepsilon_\ell) = c_0 + c_1\varphi_1(\varepsilon_\ell) + \ldots c_\nu\varphi_\nu(\varepsilon_\ell), \quad \ell = 0, \ldots, \nu;$$

- i.e., systems of order $(\nu+1) \times (\nu+1)$ in unknowns $c_0, \ldots, c_\nu$ are solved for increasing values of ν.
- The coefficients $\varphi_k(\varepsilon_\ell)$ need to be known explicitly in order to apply this method.
- The computation for $\varphi_k(\varepsilon) = \varepsilon^k$ can be carried out recursively using Richardson extrapolation [4].

Non-linear extrapolation

- As an example of a non-linear extrapolation method, the ε-algorithm [6, 7] implements a sequence-to-sequence transformation recursively;
- can be applied if the φ functions are of the form

$$\varphi_k(\varepsilon) = \varepsilon^{\beta_k} \log^{\nu_k}(\varepsilon),$$

  and if a geometric sequence is used for ε;
- but the actual form of the underlying ε-dependency does not need to be specified.

tu-logo

ur-logo

Non-linear extrapolation

- As an example of a non-linear extrapolation method, the ε-algorithm [6, 7] implements a sequence-to-sequence transformation recursively;
- can be applied if the φ functions are of the form

$$\varphi_k(\varepsilon) = \varepsilon^{\beta_k} \log^{\nu_k}(\varepsilon),$$

and if a geometric sequence is used for ε;
- but the actual form of the underlying ε-dependency does not need to be specified.

tu-logo

ur-logo

Non-linear extrapolation

- As an example of a non-linear extrapolation method, the ε-algorithm [6, 7] implements a sequence-to-sequence transformation recursively;
- can be applied if the φ functions are of the form

$$\varphi_k(\varepsilon) = \varepsilon^{\beta_k} \log^{\nu_k}(\varepsilon),$$

  and if a geometric sequence is used for ε;
- but the actual form of the underlying ε-dependency does not need to be specified.

Non-linear extrapolation

ε-algorithm table

$$
\begin{array}{ccccc}
 & \tau_{00} & & & \\
0 & & \tau_{01} & & \\
 & \tau_{10} & & \tau_{02} & \\
0 & & \tau_{11} & & \cdots \\
 & & \cdots & & \cdots \\
 & & \cdots & & \cdots \\
0 & & \tau_{\kappa-1,1} & & \cdots \\
 & \tau_{\kappa 0} & & \tau_{\kappa-1,2} & \\
0 & & \tau_{\kappa 1} & & \\
 & \tau_{\kappa+1,0} & & &
\end{array}
$$

With original sequence S_κ, for $\kappa = 0, 1, \ldots$:

$$\tau_{\kappa,-1} = 0$$

$$\tau_{\kappa 0} = S_\kappa$$

$$\tau_{\kappa,\lambda+1} = \tau_{\kappa+1,\lambda+1} + \frac{1}{\tau_{\kappa+1,\lambda} - \tau_{\kappa\lambda}}$$

Example extrapolation

$$I(\alpha)f = \int_0^1 dx_1 \int_0^1 dx_2 \, \frac{2\alpha x_2}{(x_1+x_2-1)^2+\alpha^2} = 2\arctan\frac{1}{\alpha} - \alpha\log(1+\frac{1}{\alpha^2})$$

Extrapolation table

p	$Q(10^{-p})f$			
0	0.877649149			
1	2.480743286	3.315088757		
2	**3.**029488916	3.146268404	3.141547464	
3	**3.1**25777143	3.141849664	3.141592605	3.141592651
4	**3.13**9550588	3.141610354	3.141592651	**3.14159265**7
5	**3.141**342396	3.141594001	**3.14159265**6	
6	**3.1415**63021	**3.141592**765		
7	**3.1415**89231			

Outline

1. Computational background
   - Iterated integration
   - Extrapolation

2. Hexagon reduction
   - Reduction n-dimensional N-point function
   - Triangle functions
   - Box functions

Brief overview from Binoth et al. [2]

- Representation:

$$I_N^n = \int \frac{d^n k}{i\pi^{n/2}} \frac{1}{\prod_{\ell=1}^{N}((k-r_\ell)^2 - m_\ell^2)}$$

with external momenta p_j and $r_\ell = \sum_{j=1}^{\ell} p_j$.

- The n-dimensional hexagon, pentagon and box functions ($N = 6, 5, 4$) are expressed in terms of n-dimensional triangle and $n+2$-dimensional box functions.
- In non-exceptional kinematic conditions, N-point functions with $N \geq 6$ can be expressed in terms of pentagon functions.

Brief overview from Binoth et al. [2]

- Representation:

$$I_N^n = \int \frac{d^n k}{i\pi^{n/2}} \frac{1}{\prod_{\ell=1}^{N}((k - r_\ell)^2 - m_\ell^2)}$$

  with external momenta p_j and $r_\ell = \sum_{j=1}^{\ell} p_j$.

- The n-dimensional hexagon, pentagon and box functions ($N = 6, 5, 4$) are expressed in terms of n-dimensional triangle and $n + 2$-dimensional box functions.
- In non-exceptional kinematic conditions, N-point functions with $N \geq 6$ can be expressed in terms of pentagon functions.

Brief overview from Binoth et al. [2]

- Representation:

$$I_N^n = \int \frac{d^n k}{i\pi^{n/2}} \frac{1}{\prod_{\ell=1}^{N}((k-r_\ell)^2 - m_\ell^2)}$$

  with external momenta p_j and $r_\ell = \sum_{j=1}^{\ell} p_j$.
- The n-dimensional hexagon, pentagon and box functions ($N = 6, 5, 4$) are expressed in terms of n-dimensional triangle and $n+2$-dimensional box functions.
- In non-exceptional kinematic conditions, N-point functions with $N \geq 6$ can be expressed in terms of pentagon functions.

Reduction overview

Reduction

$$I_N^n = \sum_{\kappa=1}^N B_\kappa I_{N-1,\kappa} + (N-n-1)\,\frac{\det(G)}{\det(S)}\, I_N^{n+2}, \quad \det(S) \neq 0,$$

- G is the Gram matrix, $\text{rank}(G) = \min\{4, N-1\}$ and $B_\kappa = -\sum_{\lambda=1}^N S_{\kappa\lambda}^{-1}$,
  $S_{\kappa\lambda} = -(r_\lambda - r_\kappa)^2 + m_\lambda^2 + m_\kappa^2, \quad 1 \leq \kappa, \lambda \leq N$
- hexagon I_6^n = lin. combination of six pentagon I_5^n functions, pentagon I_5^n = lin. combination of five box I_4^n fncs. $+ \; \mathcal{O}(\varepsilon)$, box I_4^n = lin. combination of four triangle I_3^n and a box I_4^{n+2}
- Infrared singularities show up in the box and triangle functions through poles in $\frac{1}{\varepsilon} = \frac{2}{4-n}$ and can be handled through sector decomposition.

tu-logo

ur-logo

Reduction overview

Reduction

$$I_N^n = \sum_{\kappa=1}^{N} B_\kappa I_{N-1,\kappa} + (N-n-1)\,\frac{\det(G)}{\det(S)}\, I_N^{n+2}, \quad \det(S) \neq 0,$$

- G is the Gram matrix, $\mathrm{rank}(G) = \min\{4, N-1\}$ and $B_\kappa = -\sum_{\lambda=1}^{N} S^{-1}_{\kappa\lambda}$, $S_{\kappa\lambda} = -(r_\lambda - r_\kappa)^2 + m_\lambda^2 + m_\kappa^2, \;\; 1 \leq \kappa, \lambda \leq N$
- hexagon I_6^n = lin. combination of six pentagon I_5^n functions, pentagon I_5^n = lin. combination of five box I_4^n fncs. + $\mathcal{O}(\varepsilon)$, box I_4^n = lin. combination of four triangle I_3^n and a box I_4^{n+2}
- Infrared singularities show up in the box and triangle functions through poles in $\frac{1}{\varepsilon} = \frac{2}{4-n}$ and can be handled through sector decomposition.

Reduction overview

Reduction

$$I_N^n = \sum_{\kappa=1}^N B_\kappa I_{N-1,\kappa} + (N-n-1)\,\frac{\det(G)}{\det(S)}\, I_N^{n+2}, \quad \det(S) \neq 0,$$

- G is the Gram matrix, $\mathrm{rank}(G) = \min\{4, N-1\}$ and
  $B_\kappa = -\sum_{\lambda=1}^N S_{\kappa\lambda}^{-1}$,
  $S_{\kappa\lambda} = -(r_\lambda - r_\kappa)^2 + m_\lambda^2 + m_\kappa^2, \quad 1 \leq \kappa, \lambda \leq N$
- hexagon I_6^n = lin. combination of six pentagon I_5^n functions,
  pentagon I_5^n = lin. combination of five box I_4^n fncs. + $\mathcal{O}(\varepsilon)$,
  box I_4^n = lin. combination of four triangle I_3^n and a box I_4^{n+2}
- Infrared singularities show up in the box and triangle functions through poles in $\frac{1}{\varepsilon} = \frac{2}{4-n}$ and can be handled through sector decomposition.

Outline

Triangle functions

- Through sector decomposition, I_3^n is split into three sector functions S_{Tri}^n and
  $S_{Tri}^4(s_1, s_2, s_3, m_1^2, m_2^2, m_3^2) = \int_0^1 dt_1 dt_2 \frac{1}{1+t_1+t_2} \frac{1}{At_2^2+Bt_2+C+i\delta}$
  where A, B and C are constant, linear and quadratic functions in t_1, respectively, and $R = B^2 - 4AC - i\delta$.
- Binoth et al. evaluate the inner integral analytically and use DQAGS from Quadpack for the outer integration. The integrand has $\sqrt{R}$ and logarithmic singularities.
- We find that we can efficiently compute the 1D×1D inner and outer integrals numerically; we used DQAGE recursively. See plots of inner 1D integrand evaluations for $S_{Tri}^{n=4}(6, 4, 1, 1, 1, 1)$ and $S_{Tri}^{n=4}(10, 4, \frac{5}{2}, 1, 1, 1)$ (matching the analytic calculation).

Triangle functions

- Through sector decomposition, I_3^n is split into three sector functions S_{Tri}^n and
  $S_{Tri}^4(s_1, s_2, s_3, m_1^2, m_2^2, m_3^2) = \int_0^1 dt_1 dt_2 \frac{1}{1+t_1+t_2} \frac{1}{At_2^2+Bt_2+C+i\delta}$
  where A, B and C are constant, linear and quadratic functions in t_1, respectively, and $R = B^2 - 4AC - i\delta$.
- Binoth et al. evaluate the inner integral analytically and use DQAGS from Quadpack for the outer integration. The integrand has $\sqrt{R}$ and logarithmic singularities.
- We find that we can efficiently compute the 1D$\times$1D inner and outer integrals numerically; we used DQAGE recursively. See plots of inner 1D integrand evaluations for $S_{Tri}^{n=4}(6, 4, 1, 1, 1, 1)$ and $S_{Tri}^{n=4}(10, 4, \frac{5}{2}, 1, 1, 1)$ (matching the analytic calculation).

Triangle functions

- Through sector decomposition, I_3^n is split into three sector functions S_{Tri}^n and
  $S_{Tri}^4(s_1, s_2, s_3, m_1^2, m_2^2, m_3^2) = \int_0^1 dt_1 dt_2 \frac{1}{1+t_1+t_2} \frac{1}{At_2^2+Bt_2+C+i\delta}$
  where A, B and C are constant, linear and quadratic functions in t_1, respectively, and $R = B^2 - 4AC - i\delta$.
- Binoth et al. evaluate the inner integral analytically and use DQAGS from Quadpack for the outer integration. The integrand has $\sqrt{R}$ and logarithmic singularities.
- We find that we can compute the efficiently compute the $1D\times 1D$ inner and outer integrals numerically; we used DQAGE recursively. See plots of inner 1D integrand evaluations for $S_{Tri}^{n=4}(6, 4, 1, 1, 1, 1)$ and $S_{Tri}^{n=4}(10, 4, \frac{5}{2}, 1, 1, 1)$ (matching the analytic calculation).

Plots of inner 1D integrand evaluations

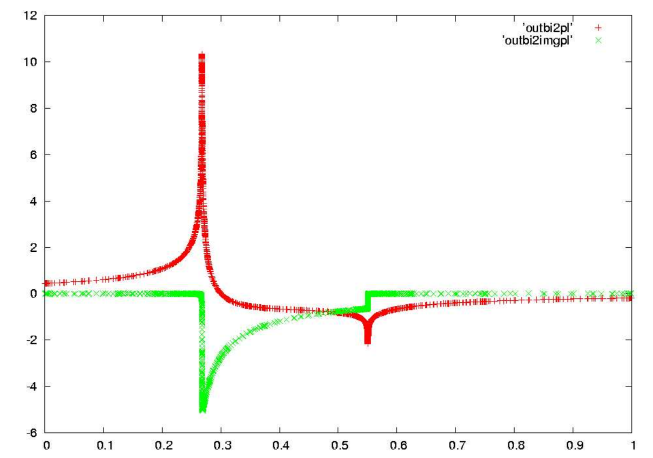

Figure: Inner integrand evaluation by DQAGE for $S^{n=4}_{Tri}(6, 4, 1, 1, 1, 1)$

Plots of inner 1D integrand evaluations

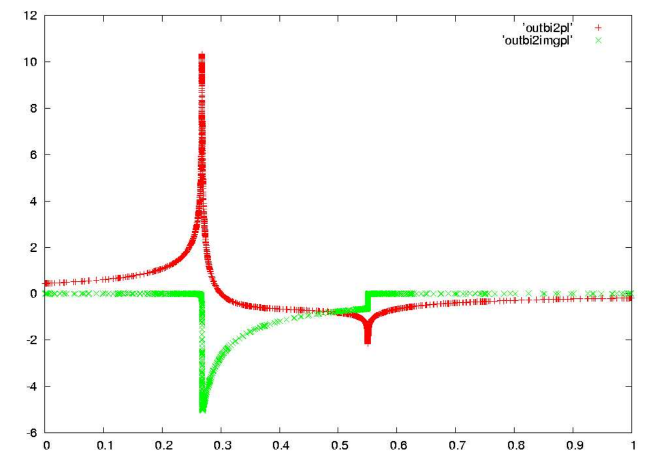

Figure: Inner integrand evaluation by DQAGE for $S_{Tri}^{n=4}(10, 4, \frac{5}{2}, 1, 1, 1)$

Sample program of 1D×1D outer integrand function

```
      double precision function fx(x)
      implicit real*8(a-h,o-z)
      parameter(nw = 1000)
      dimension alist(nw),blist(nw),elist(nw),rlist(nw),iord(nw)
      common/wrk/epsa,epsr,lim,keyy
      common/limits/ay,by
      common/args/xx
      common/flags/iflagy
      external fy
      epsabs = epsa
      epsrel = epsr
      limit = lim
      xx = x
C Integration in y direction
      call dqagey(fy,ay,by,epsabs,epsrel,keyy,limit,result,abserr,neval,
     *      ier,alist,blist,rlist,elist,iord,last)
      if(ier.ne.0) iflagy = iflagy+1
      fx = result
      return
```

Sample program of 1D×1D inner integrand function

```
      double precision function fy(y)
      implicit real*8(a-h,o-z)
      common/pars/eps,sqeps,dm1,dm2,dm3,s1,s2,s3
      common/args/xx
      common/icnt/dkount
      dkount = dkount+1.d0
      aa = dm2
      bb = (dm1+dm2-s2)*xx+dm2+dm3-s3
      cc = dm1*xx*xx+(dm1+dm3-s1)*xx+dm3
      d = aa*y*y+bb*y+cc
      denom = d*d+sqeps
C Real part
C     fy = d/denom/(1+xx+y)
C Imaginary part
      fy = -eps/denom/(1+xx+y)
      return
      end
```

Outline

1. Computational background
   - Iterated integration
   - Extrapolation
2. Hexagon reduction
   - Reduction n-dimensional N-point function
   - Triangle functions
   - Box functions

Box functions

- Recall,
  box I_4^n = lin. combination of four triangle I_3^n and box I_4^{n+2}.
- I_4^{n+2} is split into four sector integrals of the form S_{Box}^{n+2};
- $S_{Box}^{n=6}(s_{12}, s_{23}, s_1, s_2, s_3, s_4, m_1^2, m_2^2, m_3^2, m_4^2)$
  $= \int_0^1 dt_1 dt_2 dt_3 \frac{1}{(1+t_1+t_2+t_3)^2} \frac{1}{At_2^2+Bt_2+C-i\delta}$
  where A, B and C are constant, linear and quadratic functions in t_1, t_2 and $R = B^2 - 4AC + i\delta$.
- Binoth et al. evaluate the inner integral analytically and use DCUHRE [1] for the outer 2D integration.

Box functions

- Recall,
  box I_4^n = lin. combination of four triangle I_3^n and box I_4^{n+2}.
- I_4^{n+2} is split into four sector integrals of the form S_{Box}^{n+2};
- $S_{Box}^{n=6}(s_{12}, s_{23}, s_1, s_2, s_3, s_4, m_1^2, m_2^2, m_3^2, m_4^2)$
  $= \int_0^1 dt_1 dt_2 dt_3 \frac{1}{(1+t_1+t_2+t_3)^2} \frac{1}{At_2^2+Bt_2+C-i\delta}$
  where A, B and C are constant, linear and quadratic functions in t_1, t_2 and $R = B^2 - 4AC + i\delta$.
- Binoth et al. evaluate the inner integral analytically and use DCUHRE [1] for the outer 2D integration.

tu-logo

ur-logo

Box functions

- Recall,
  box I_4^n = lin. combination of four triangle I_3^n and box I_4^{n+2}.
- I_4^{n+2} is split into four sector integrals of the form S_{Box}^{n+2};
- $S_{Box}^{n=6}(s_{12}, s_{23}, s_1, s_2, s_3, s_4, m_1^2, m_2^2, m_3^2, m_4^2)$
  $= \int_0^1 dt_1 dt_2 dt_3 \frac{1}{(1+t_1+t_2+t_3)^2} \frac{1}{At_2^2+Bt_2+C-i\delta}$
  where A, B and C are constant, linear and quadratic functions in t_1, t_2 and $R = B^2 - 4AC + i\delta$.
- Binoth et al. evaluate the inner integral analytically and use DCUHRE [1] for the outer 2D integration.

Box functions

- Recall,
  box I_4^n = lin. combination of four triangle I_3^n and box I_4^{n+2}.
- I_4^{n+2} is split into four sector integrals of the form S_{Box}^{n+2};
- $S_{Box}^{n=6}(s_{12}, s_{23}, s_1, s_2, s_3, s_4, m_1^2, m_2^2, m_3^2, m_4^2)$
  $= \int_0^1 dt_1 dt_2 dt_3 \frac{1}{(1+t_1+t_2+t_3)^2} \frac{1}{At_2^2+Bt_2+C-i\delta}$
  where A, B and C are constant, linear and quadratic functions in t_1, t_2 and $R = B^2 - 4AC + i\delta$.
- Binoth et al. evaluate the inner integral analytically and use DCUHRE [1] for the outer 2D integration.

Box functions

- The analytic integrand evaluation has a complicated singularity structure and the 2D integration is probematic. Binoth et al. combine the numeric integration by DCUHRE with a Monte-Carlo integration in the vicinity of singular behavior.
- It is mentioned that DCUHRE was used with a workspace limit of 350MB to allow a maximum of 1.5 10^9 2D function evaluations.
- We find that we can efficiently compute the 3D integral recursively with DQAGE from Quadpack [5] as a 1D$\times$1D$\times$1D integral.

Box functions

- The analytic integrand evaluation has a complicated singularity structure and the 2D integration is probematic. Binoth et al. combine the numeric integration by DCUHRE with a Monte-Carlo integration in the vicinity of singular behavior.
- It is mentioned that DCUHRE was used with a workspace limit of 350MB to allow a maximum of 1.5 10^9 2D function evaluations.
- We find that we can efficiently compute the 3D integral recursively with DQAGE from Quadpack [5] as a 1D×1D×1D integral.

Box functions

- The analytic integrand evaluation has a complicated singularity structure and the 2D integration is probematic. Binoth et al. combine the numeric integration by DCUHRE with a Monte-Carlo integration in the vicinity of singular behavior.
- It is mentioned that DCUHRE was used with a workspace limit of 350MB to allow a maximum of 1.5 10^9 2D function evaluations.
- We find that we can efficiently compute the 3D integral recursively with DQAGE from Quadpack [5] as a $1D\times1D\times1D$ integral.

Conclusions

- It was shown that recursive numerical integration and extrapolation can be used (as efficient computational building blocks) to perform the reduction numerically, down from the level of box and triangle integrals.
- An $n-d$ pentagon function is split into five $n-d$ box functions;
  each of those into four $n-d$ triangle functions and an $(n+2)-d$ box function.
  Thus the $n-d$ pentagon is split into 20 $n-d$ triangle functions (10 different ones through symmetry) and five $(n+2)-d$ box functions.
- The $n-d$ hexagon is split into 20 $n-d$ triangle functions and 15 $(n+2)-d$ box functions. We evaluated these pieces numerically even in the presence of singularities in the interior of the integration domain.

Conclusions

- It was shown that recursive numerical integration and extrapolation can be used (as efficient computational building blocks) to perform the reduction numerically, down from the level of box and triangle integrals.
- An $n-d$ pentagon function is split into five $n-d$ box functions;
  each of those into four $n-d$ triangle functions and an $(n+2)-d$ box function.
  Thus the $n-d$ pentagon is split into 20 $n-d$ triangle functions (10 different ones through symmetry) and five $(n+2)-d$ box functions.
- The $n-d$ hexagon is split into 20 $n-d$ triangle functions and 15 $(n+2)-d$ box functions. We evaluated these pieces numerically even in the presence of singularities in the interior of the integration domain.

Conclusions

- It was shown that recursive numerical integration and extrapolation can be used (as efficient computational building blocks) to perform the reduction numerically, down from the level of box and triangle integrals.
- An $n - d$ pentagon function is split into five $n - d$ box functions;
  each of those into four $n - d$ triangle functions and an $(n + 2) - d$ box function.
  Thus the $n - d$ pentagon is split into 20 $n - d$ triangle functions (10 different ones through symmetry) and five $(n + 2) - d$ box functions.
- The $n - d$ hexagon is split into 20 $n - d$ triangle functions and 15 $(n + 2) - d$ box functions. We evaluated these pieces numerically even in the presence of singularities in the interior of the integration domain.

Conclusions

- The current strategy needs improvements, e.g., with respect to accuracy control (e.g., automatic adjusting of the number of extrapolations for required accuracy).
- Separating infinite and finite parts in case of IR divergences can be incorporated in a transparent way; is also being done by other authors.
- The integration method is suited to handling non-scalar cases numerically in a flexible manner.

Conclusions

- The current strategy needs improvements, e.g., with respect to accuracy control (e.g., automatic adjusting of the number of extrapolations for required accuracy).
- Separating infinite and finite parts in case of IR divergences can be incorporated in a transparent way; is also being done by other authors.
- The integration method is suited to handling non-scalar cases numerically in a flexible manner.

Conclusions

- The current strategy needs improvements, e.g., with respect to accuracy control (e.g., automatic adjusting of the number of extrapolations for required accuracy).
- Separating infinite and finite parts in case of IR divergences can be incorporated in a transparent way; is also being done by other authors.
- The integration method is suited to handling non-scalar cases numerically in a flexible manner.

Bibliography

Berntsen, J., Espelid, T. O., and Genz, A.
Algorithm 698: DCUHRE-an adaptive multidimensional integration routine for a vector of integrals.
ACM Trans. Math. Softw. 17 (1991), 452–456.

Binoth, T., Heinrich, G., and Kauer, N.
A numerical evaluation of the scalar hexagon integral in the physical region.
hep-ph/0210023.

Brezinski, C.
A general extrapolation algorithm.
Numerische Mathematik 35 (1980), 175–187.

Davis, P. J., and Rabinowitz, P.

Methods of Numerical Integration.
Academic Press, New York, 1984.

PIESSENS, R., DE DONCKER, E., ÜBERHUBER, C. W., AND KAHANER, D. K.
QUADPACK, A Subroutine Package for Automatic Integration.
Springer Series in Computational Mathematics. Springer-Verlag, 1983.

SHANKS, D.
Non-linear transformations of divergent and slowly convergent sequences.
J. Math. and Phys. 34 (1955), 1–42.

WYNN, P.
On a device for computing the $e_m(s_n)$ transformation.
Mathematical Tables and Aids to Computing 10 (1956), 91–96.