the MSR-INRIA Joint Centre

Jean-Jacques Lévy

June 2, 2008

CENTRE DE RECHERCHE COMMUN
INRIA MICROSOFT RESEARCH

Plan

1. Context
2. Track A
   - Math. Components
   - Security
   - TLA+
3. Track B
   - DDMF
   - ReActivity
   - Adaptative search
   - Image & video mining

Context

CENTRE DE RECHERCHE COMMUN

INRIA MICROSOFT RESEARCH

Politics

INRIA

Gilles Kahn

Michel Cosnard

Michel Bidoit
Bruno Sportisse

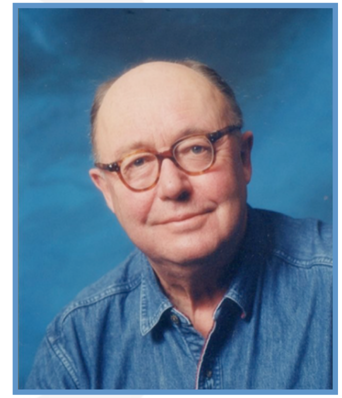
Eric Boustouller
Stephen Emmott
Gérard Giraudon
Gérard Huet
Jean Vuillemin

Joint Centre

J.-J. Lévy

Stephen Emmott
Malik Ghallab
Claude Puech
Ken Wood

MSR Cambridge

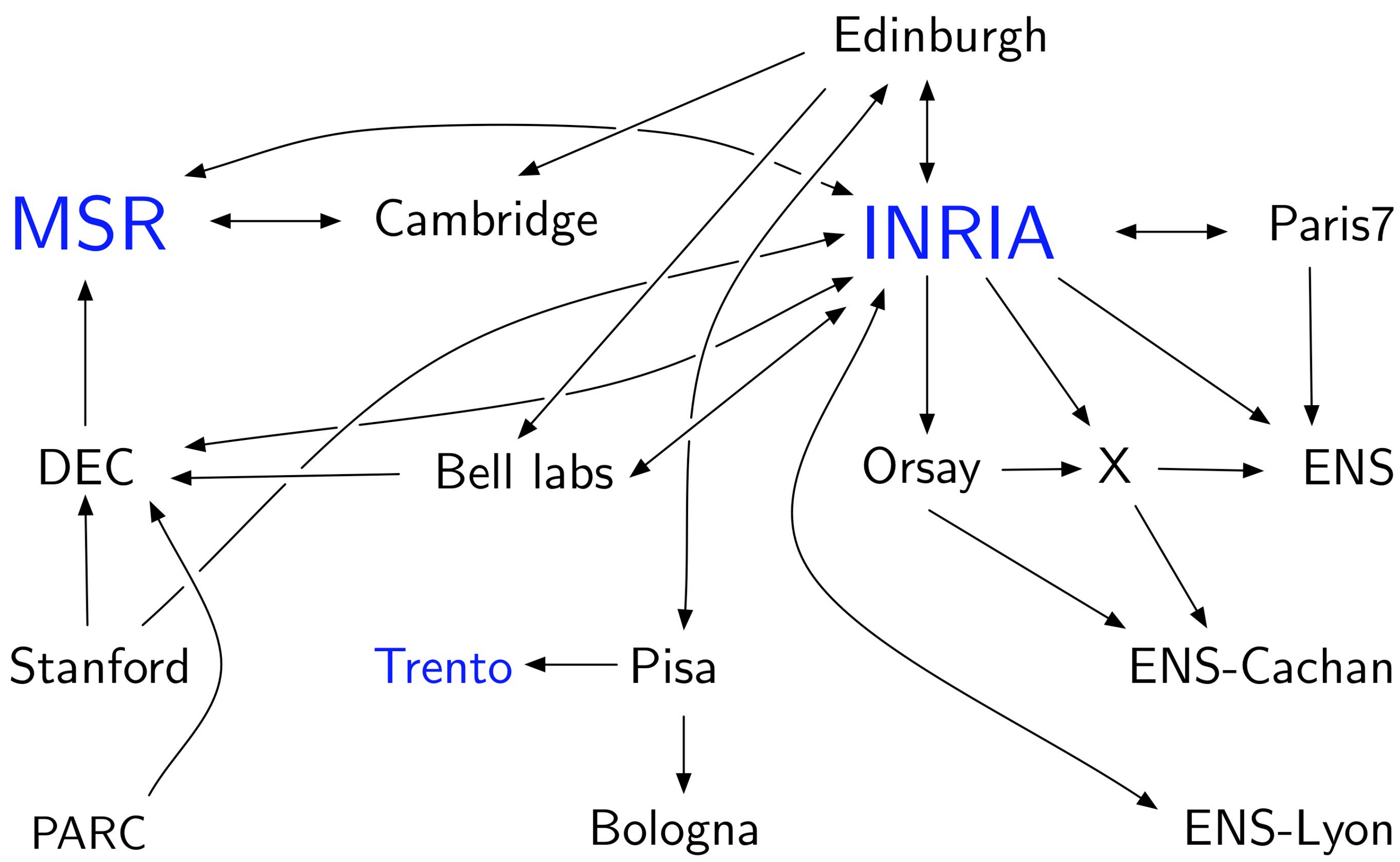


Roger Needham

Andrew Herbert

Marc Jalabert
Bernard Ourghanlian

Long cooperation among researchers



Organization

a rather complex system

- **7 research projects (in two tracks)**
- **12 resident researchers**
- **non permanent researchers funded by the Joint Centre**
- **permanent researchers paid by INRIA or MSR**
- **operational support by INRIA Saclay**
- **1 system manager** (Guillaume Rousse, INRIA Saclay)
- **1 administrative assistant** (Martine Thirion, Joint Centre)
- **1 deputy director** (Pierre-Louis Xech, MS France)
- **active support from MS France**

People

PhD Students

- Alexandro ARBALAEZ
- Alvaro FIALHO
- Francois GARILLOT
- Sidi OULD BIHA
- Iona PASCA
- Arnaud SPIVAK
- Nicolas MASSON
- Nathalie HENRY
- Nataliya GUTS
- Santiago ZANELLA

Post Docs

- Stéphane LEROUX
- Guillaume MELQUIOND
- Roland ZUMKELLER
- Assia MAHBOUBI (*)
- Aurélien TABARD
- Catherine LEDONTAL
- Niklas ELMQVIST
- Gurvan LE GUERNIC
- Eugen ZALINESCU
- Ricardo CORIN (*)
- Tamara REZK (*)

Interns

- Jorge Martin PEREZ-ZERPA
- Sébastien MIGNOT
- Fabien TEYTAUD
- Alexandre BENOIT
- Pratik PODDAR
- Sean McLAUGHLIN
- Etienne MIRET
- Enrico TASSI
- Fei LI
- Yoann COLDFLY
- Jérémy PLANUL

(*) Now on permanent INRIA position

Track A

Software Security

Trustworthy Computing

CENTRE DE RECHERCHE COMMUN
INRIA MICROSOFT RESEARCH

Mathematical components

Georges Gonthier, MSRC
Assia Mahboubi, INRIA Saclay/LIX
Andrea Asperti, Bologna
Y. Bertot, L. Rideau, L. Théry, Sidi Ould Biha, Iona Pasca, INRIA Sophia
François Garillot, MSR-INRIA (PhD)
Guillaume Melquiond, MSR-INRIA (postdoc)
Stéphane le Roux, MSR-INRIA (postdoc)
Benjamin Werner, INRIA Saclay/LIX,
Roland Zumkeller, LIX (PhD)

Computational proofs

– computer assistance for long formal proofs.
– reflection of computations into Coq-logic: ssreflect.

4-color

Appel-Haken

finite groups

Feit-Thompson

Kepler

Hales

```
Section R_props.

(* The ring axioms, and some useful basic corollaries. *)

Hypothesis mult1x : forall x, 1 * x = x.
Hypothesis mult0x : forall x : R, 0 * x = 0.
Hypothesis plus0x : forall x : R, 0 + x = x.
Hypothesis minusxx : forall x : R, x - x = 0.
Hypothesis plusA : forall x1 x2 x3 : R, x1 + (x2 + x3) = x1 + x2 + x3.
Hypothesis plusC : forall x1 x2 : R, x1 + x2 = x2 + x1.
Hypothesis multA : forall x1 x2 x3 : R, x1 * (x2 * x3) = x1 * x2 * x3.
Hypothesis multC : forall x1 x2 : R, x1 * x2 = x2 * x1.
Hypothesis distrR : forall x1 x2 x3 : R, (x1 + x2) * x3 = x1 * x3 + x2 * x3.

Lemma plusCA : forall x1 x2 x3 : R, x1 + (x2 + x3) = x2 + (x1 + x3).
Proof. move=> *; rewrite !plusA; congr (_ + _); exact: plusC. Qed.

Lemma multCA : forall x1 x2 x3 : R, x1 * (x2 * x3) = x2 * (x1 * x3).
Proof. move=> *; rewrite !multA; congr (_ * _); exact: multC. Qed.

Lemma distrL : forall x1 x2 x3 : R, x1 * (x2 + x3) = x1 * x2 + x1 * x3.
Proof. by move=> x1 x2 x3; rewrite !(multC x1) distrR. Qed.

Lemma oppK : involutive opp.
Proof.
by move=> x; rewrite -{2}[x]plus0x -(minusxx (- x)) plusC plusA minusxx plus0x.
Qed.

Lemma multm1x : forall x, -1 * x = -x.
Proof.
move=> x; rewrite -[_ * x]plus0x -(minusxx x) -{1}[x]mult1x plusC plusCA plusA.
by rewrite -distrR minusxx mult0x plus0x.
Qed.

Lemma mult_opp : forall x1 x2 : R, (- x1) * x2 = - (x1 * x2).
Proof. by move=> *; rewrite -multm1x -multA multm1x. Qed.

Lemma opp_plus : forall x1 x2 : R, - (x1 + x2) = - x1 - x2.
Proof.
by move=> x1 x2; rewrite -multm1x multC distrR -!(multC -1) !multm1x.
Qed.

Lemma RofSnE : forall n, RofSn n = n + 1.
Proof. by elim=> /= [|_ -> //]; rewrite plus0x. Qed.

Lemma Raddn : forall m n, (m + n)%N = m + n :> R.
Proof.
move=> m n; elim: m => /= [|m IHm]; first by rewrite plus0x.
by rewrite !RofSnE IHm plusC plusCA plusA.
Qed.

Lemma Rsubn : forall m n, m >= n -> (m - n)%N = m - n :> R.
 -(DOS)--  determinant.v   42% (709,42)    (coq)
```

l i n e a r a l g e b r a

CENTRE DE RECHERCHE COMMUN
INRIA MICROSOFT RESEARCH

```
Section R_props.

(* The ring axioms, and some useful basic corollaries. *)

Hypothesis mult1x : forall x, 1 * x = x.
Hypothesis mult0x : forall x : R, 0 * x = 0.
Hypothesis plus0x : forall x : R, 0 + x = x.
Hypothesis minusxx : forall x : R, x - x = 0.
Hypothesis plusA : forall x1 x2 x3 : R, x1 + (x2 + x3) = x1 + x2 + x3.
Hypothesis plusC : forall x1 x2 : R, x1 + x2 = x2 + x1.
Hypothesis multA : forall x1 x2 x3 : R, x1 * (x2 * x3) = x1 * x2 * x3.
Hypothesis multC : forall x1 x2 : R, x1 * x2 = x2 * x1.
Hypothesis distrR : forall x1 x2 x3 : R, (x1 + x2) * x3 = x1 * x3 + x2 * x3.

Lemma plusCA : forall x1 x2 x3 : R, x1 + (x2 + x3) = x2 + (x1 + x3).
Proof. move=> *; rewrite !plusA; congr (_ + _); exact: plusC. Qed.

Lemma multCA : forall x1 x2 x3 : R, x1 * (x2 * x3) = x2 * (x1 * x3).
Proof. move=> *; rewrite !multA; congr (_ * _); exact: multC. Qed.

Lemma distrL : forall x1 x2 x3 : R, x1 * (x2 + x3) = x1 * x2 + x1 * x3.
Proof. by move=> x1 x2 x3; rewrite !(multC x1) distrR. Qed.

Lemma oppK : involutive opp.
Proof.
by move=> x; rewrite -{2}[x]plus0x -(minusxx (- x)) plusC plusA minusxx plus0x.
Qed.

Lemma multm1x : forall x, -1 * x = -x.
Proof.
move=> x; rewrite -[_ * x]plus0x -(minusxx x) -{1}[x]mult1x plusC plusCA plusA.
by rewrite -distrR minusxx mult0x plus0x.
Qed.
```

```
Lemma Rsubn : forall m n, m >= n -> (m - n)%N = m - n :> R.
Proof.
move=> m n; move/leq_add_sub=> Dm.
by rewrite -{2}Dm Raddn -plusA plusCA minusxx plusC plus0x.
Qed.

Lemma Rmuln : forall m n, (m * n)%N = m * n :> R.
Proof.
move=> m n; elim: m => /= [|m IHm]; first by rewrite mult0x.
by rewrite Raddn RofSnE IHm distrR mult1x plusC.
Qed.

Lemma RexpSnE : forall x n, RexpSn x n = x ^ n * x.
Proof. by move=> x; elim=> /= [|_ -> //]; rewrite mult1x. Qed.

Lemma mult_exp : forall x1 x2 n, (x1 * x2) ^ n = x1 ^ n * x2 ^ n.
Proof.
by move=> x1 x2; elim=> //= n IHn; rewrite !RexpSnE IHn -!multA (multCA x1).
Qed.

Lemma exp_addn : forall x n1 n2, x ^ (n1 + n2) = x ^ n1 * x ^ n2.
Proof.
move=> x n1 n2; elim: n1 => /= [|n1 IHn]; first by rewrite mult1x.
by rewrite !RexpSnE IHn multC multCA multA.
Qed.

Lemma Rexpn : forall m n, (m ^ n)%N = m ^ n :> R.
Proof. by move=> m; elim=> //= n IHn; rewrite Rmuln RexpSnE IHn multC. Qed.

Lemma exp0n : forall n, 0 < n -> 0 ^ n = 0.
Proof. by move=> [|[|n]] //= _; rewrite multC mult0x. Qed.

Lemma exp1n : forall n, 1 ^ n = 1.
Proof. by elim=> //= n IHn; rewrite RexpSnE IHn mult1x. Qed.

Lemma exp_muln : forall x n1 n2, x ^ (n1 * n2) = (x ^ n1) ^ n2.
Proof.
move=> x n1 n2; rewrite mulnC; elim: n2 => //= n2 IHn.
by rewrite !RexpSnE exp_addn IHn multC.
Qed.

Lemma sign_odd : forall n, (-1) ^ odd n = (-1) ^ n.
Proof.
move=> n; rewrite -{2}[n]odd_double_half addnC double_mul2 exp_addn exp_muln.
by rewrite /= multm1x oppK exp1n mult1x.
Qed.

Lemma sign_addb : forall b1 b2, (-1) ^ (b1 (+) b2) = (-1) ^ b1 * (-1) ^ b2.
Proof. by do 2!case; rewrite //= ?multm1x ?mult1x ?oppK. Qed.

Lemma sign_permM : forall d (s t : permType d),
-(DOS)--  determinant.v   45% (760,61)    (coq)
```

l i n e a r

a l g e b r a

CENTRE DE RECHERCHE COMMUN

INRIA
MICROSOFT RESEARCH

```
rewrite isum0 ?plus0x // => i'; rewrite andbT; move/negbET->; exact: mult0x.
Qed.

Lemma matrix_transpose_mul : forall m n p (A : M_(m, n)) (B : M_(n, p)),
   \^t (A *m B) =m \^t B *m \^t A.
Proof. split=> k i; apply: eq_isumR => j _; exact: multC. Qed.

Lemma matrix_multx1 : forall m n (A : M_(m, n)), A *m \1m =m A.
Proof.
move=> m n A; apply: matrix_transpose_inj.
by rewrite matrix_transpose_mul matrix_transpose_unit matrix_mult1x.
Qed.

Lemma matrix_distrR : forall m n p (A1 A2 : M_(m, n)) (B : M_(n, p)),
  (A1 +m A2) *m B =m A1 *m B +m A2 *m B.
Proof.
move=> m n p A1 A2 B; split=> i k /=; rewrite -isum_plus.
by apply: eq_isumR => j _; rewrite -distrR.
Qed.

Lemma matrix_distrL : forall m n p (A : M_(m, n)) (B1 B2 : M_(n, p)),
  A *m (B1 +m B2) =m A *m B1 +m A *m B2.
Proof.
move=> m n p A B1 B2; apply: matrix_transpose_inj.
rewrite matrix_transpose_plus !matrix_transpose_mul.
by rewrite -matrix_distrR -matrix_transpose_plus.
Qed.

Lemma matrix_multA : forall m n p q
   (A : M_(m, n)) (B : M_(n, p)) (C : M_(p, q)),
  A *m (B *m C) =m A *m B *m C.
Proof.
move=> m n p q A B C; split=> i l /=.
transitivity (\sum_(k) (\sum_(j) (A i j * B j k * C k l))).
  rewrite exchange_isum; apply: eq_isumR => j _; rewrite isum_distrL.
  by apply: eq_isumR => k _; rewrite multA.
by apply: eq_isumR => j _; rewrite isum_distrR.
Qed.

Lemma perm_matrixM : forall n (s t : S_(n)),
  perm_matrix (s * t)%G =m perm_matrix s *m perm_matrix t.
Proof.
move=> n; split=> i j /=; rewrite (isumD1 (s i)) // set11 mult1x -permM.
rewrite isum0 => [|j']; first by rewrite plusC plus0x.
by rewrite andbT; move/negbET->; rewrite mult0x.
Qed.

Lemma matrix_trace_plus : forall n (A B : M_(n)), \tr (A +m B) = \tr A + \tr B.
Proof. by move=> n A B; rewrite -isum_plus. Qed.

Lemma matrix_trace_scale : forall n x (A : M_(n)), \tr (x *sm A) = x * \tr A.
Proof. by move=> *; rewrite isum_distrL. Qed.
-(DOS)--  determinant.v   77% (1190,48)   (coq)
```

linear algebra

CENTRE DE RECHERCHE COMMUN
INRIA MICROSOFT RESEARCH

```
(* And now, finally, the title feature. *)

Lemma determinant_multilinear : forall n (A B C : M_(n)) i0 b c,
    row i0 A =m b *sm row i0 B +m c *sm row i0 C ->
    row' i0 B =m row' i0 A -> row' i0 C =m row' i0 A ->
  \det A = b * \det B + c * \det C.
Proof.
move=> n A B C i0 b c ABC.
move/matrix_eq_rem_row=> BA; move/matrix_eq_rem_row=> CA.
rewrite !isum_distrL -isum_plus; apply: eq_isumR => s _.
rewrite -!(multCA (_ ^ s)) -distrL; congr (_ * _).
rewrite !(@iprodD1 _ i0 (setA _)) // (matrix_eq_row ABC) distrR !multA.
by congr (_ * _ + _ * _); apply: eq_iprodR => i;
   rewrite andbT => ?; rewrite ?BA ?CA.
Qed.

Lemma alternate_determinant : forall n (A : M_(n)) i1 i2,
  i1 != i2 -> A i1 =1 A i2 -> \det A = 0.
Proof.
move=> n A i1 i2 Di12 A12; pose r := I_(n).
pose t := transp i1 i2; pose tr s := (t * s)%G.
have trK : involutive tr by move=> s; rewrite /tr mulgA transp2 mul1g.
have Etr: forall s, odd_perm (tr s) = even_perm s.
  by move=> s; rewrite odd_permM odd_transp Di12.
rewrite /(\det _) (isumID (@even_perm r)) /=; set S1 := \sum_(in _) _.
rewrite -{2}(minusxx S1); congr (_ + _); rewrite {}/S1 -isum_opp.
rewrite (reindex_isum tr); last by exists tr.
symmetry; apply: eq_isum => [s | s seven]; first by rewrite negbK Etr.
rewrite -multm1x multA Etr seven (negbET seven) multm1x; congr (_ * _).
rewrite (reindex_iprod t); last by exists (t : _ -> _) => i _; exact: transpK.
apply: eq_iprodR => i _; rewrite permM /t.
by case: transpP => // ->; rewrite A12.
Qed.

Lemma determinant_transpose : forall n (A : M_(n)), \det (\^t A) = \det A.
Proof.
move=> n A; pose r := I_(n); pose ip p : permType r := p^-1.
rewrite /(\det _) (reindex_isum ip) /=; last first.
  by exists ip => s _; rewrite /ip invgK.
apply: eq_isumR => s _; rewrite odd_permV /= (reindex_iprod s).
  by congr (_ * _); apply: eq_iprodR => i _; rewrite permK.
by exists (s^-1 : _ -> _) => i _; rewrite ?permK ?permKv.
Qed.

Lemma determinant_perm : forall n s, \det (@perm_matrix n s) = (-1) ^ s.
Proof.
move=> n s; rewrite /(\det _) (isumD1 s) //.
rewrite iprod1 => [i _]; last by rewrite /= set11.
rewrite isum0 => [t Dst]; first by rewrite plusC plus0x multC mult1x.
case: (pickP (fun i => s i != t i)) => [i ist | Est].
  by rewrite (iprodD1 i) // multCA /= (negbET ist) mult0x.
move: Dst; rewrite andbT; case/eqP.
-(DOS)--  determinant.v    81% (1256,4)    (coq)
```

l
i
n
e
a
r

a
l
g
e
b
r
a

CENTRE DE RECHERCHE
COMMUN

INRIA
MICROSOFT RESEARCH

```
Lemma determinant1 : forall n, \det (unit_matrix n) = 1.
Proof.
move=> n; have:= @determinant_perm n 1%G; rewrite odd_perm1 => /= <-.
apply: determinant_extensional; symmetry; exact: perm_matrix1.
Qed.

Lemma determinant_scale : forall n x (A : M_(n)),
  \det (x *sm A) = x ^ n * \det A.
Proof.
move=> n x A; rewrite isum_distrL; apply: eq_isumR => s _.
by rewrite multCA iprod_mult iprod_id card_ordinal.
Qed.

Lemma determinantM : forall n (A B : M_(n)), \det (A *m B) = \det A * \det B.
Proof.
move=> n A B; rewrite isum_distrR.
pose AB (f : F_(n)) (s : S_(n)) i := A i (f i) * B (f i) (s i).
transitivity (\sum_(f) \sum_(s : S_(n)) (-1) ^ s * \prod_(i) AB f s i).
  rewrite exchange_isum; apply: eq_isumR => s _.
  by rewrite -isum_distrL distr_iprodA_isumA.
rewrite (isumID (fun f => uniq (fval f))) plusC isum0 ?plus0x => /= [If Uf].
  rewrite (reindex_isum (fun s => val (pval s))); last first.
    have s0 : S_(n) := 1%G; pose uf (f : F_(n)) := uniq (fval f).
    pose pf f := if insub uf f is Some s then Perm s else s0.
    exists pf => /= f Uf; rewrite /pf (insubT uf Uf) //; exact: eq_fun_of_perm.
  apply: eq_isum => [s|s _]; rewrite ?(valP (pval s)) // isum_distrL.
  rewrite (reindex_isum (mulg s)); last first.
    by exists (mulg s^-1) => t; rewrite ?mulKgv ?mulKg.
  apply: eq_isumR => t _; rewrite iprod_mult multA multCA multA multCA multA.
  rewrite -sign_permM; congr (_ * _); rewrite (reindex_iprod s^-1); last first.
    by exists (s : _ -> _) => i _; rewrite ?permK ?permKv.
  by apply: eq_iprodR => i _; rewrite permM permKv ?set11 // -{3}[i](permKv s).
transitivity (\det (\matrix_(i, j) B (f i) j) * \prod_(i) A i (f i)).
  rewrite multC isum_distrL; apply: eq_isumR=> s _.
  by rewrite multCA iprod_mult.
suffices [i1 [i2 Ef12 Di12]]: exists i1, exists2 i2, f i1 = f i2 & i1 != i2.
  by rewrite (alternate_determinant Di12) ?mult0x => //= j; rewrite Ef12.
pose ninj i1 i2 := (f i1 == f i2) && (i1 != i2).
case: (pickP (fun i1 => ~~ set0b (ninj i1))) => [i1| injf].
  by case/set0Pn=> i2; case/andP; move/eqP; exists i1; exists i2.
case/(perm_uniqP f): Uf => i1 i2; move/eqP=> Dfi12; apply/eqP.
by apply/idPn=> Di12; case/set0Pn: (injf i1); exists i2; apply/andP.
Qed.

(* And now, the Laplace formula. *)

Definition cofactor n (A : M_(n)) (i j : I_(n)) :=
   (-1) ^ (val i + val j) * \det (row' i (col' j A)).

(* Same bug as determinant
Add Morphism cofactor with
-(DOS)-- determinant.v   85% (1284,0)    (coq)
```

l i n e a r

a l g e b r a

CENTRE DE RECHERCHE COMMUN

INRIA MICROSOFT RESEARCH

```
Lemma determinant1 : forall n, \det (unit_matrix n) = 1.
Proof.
move=> n; have:= @determinant_perm n 1%G; rewrite odd_perm1 => /= <-.
apply: determinant_extensional; symmetry; exact: perm_matrix1.
Qed.

Lemma determinant_scale : forall n x (A : M_(n)),
  \det (x *sm A) = x ^ n * \det A.
Proof.
move=> n x A; rewrite isum_distrL; apply: eq_isumR => s _.
by rewrite multCA iprod_mult iprod_id card_ordinal.
Qed.

Lemma determinantM : forall n (A B : M_(n)), \det (A *m B) = \det A * \det B.
Proof.
move=> n A B; rewrite isum_distrR.
pose AB (f : F_(n)) (s : S_(n)) i := A i (f i) * B (f i) (s i).
transitivity (\sum_(f) \sum_(s : S_(n)) (-1) ^ s * \prod_(i) AB f s i).
  rewrite exchange_isum; apply: eq_isumR => s _.
  by rewrite -isum_distrL distr_iprodA_isumA.
rewrite (isumID (fun f => uniq (fval f))) plusC isum0 ?plus0x => /= [|f Uf].
  rewrite (reindex_isum (fun s => val (pval s))); last first.
    have s0 : S_(n) := 1%G; pose uf (f : F_(n)) := uniq (fval f).
    pose pf f := if insub uf f is Some s then Perm s else s0.
    exists pf => /= f Uf; rewrite /pf (insubT uf Uf) //; exact: eq_fun_of_perm.
  apply: eq_isum => [s|s _]; rewrite ?(valP (pval s)) // isum_distrL.
  rewrite (reindex_isum (mulg s)); last first.
    by exists (mulg s^-1) => t; rewrite ?mulKgv ?mulKg.
```

Secure Distributed Computations and their Proofs

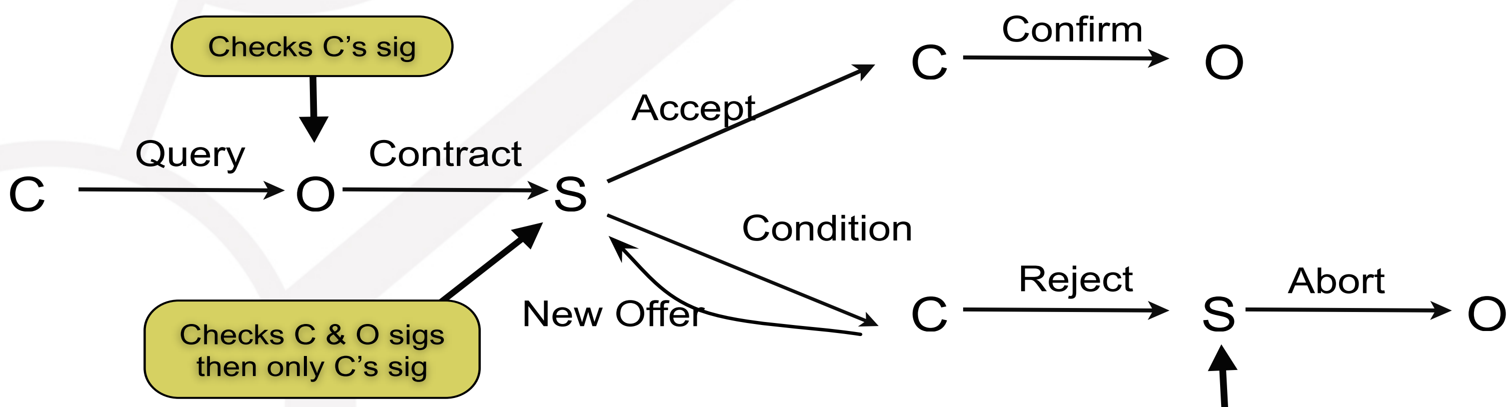
Cédric Fournet, MSRC
Karthik Bhargavan, MSRC
Ricardo Corin, INRIA Rocq.
Pierre-Malo Deniélou, INRIA Rocq.
G. Barthe, B. Grégoire, S. Zanella, INRIA Sophia
James Leifer, INRIA Rocq.
Jean-Jacques Lévy, INRIA Rocq.
Tamara Rezk, INRIA Sophia
Francesco Zappa Nardelli, INRIA Rocq.
Nataliya Guts, MSR-INRIA (PhD)
Jérémy Planul, MSR-INRIA (intern)

Distributed computations + Security

- programming with secured communications
- certified compiler from high-level primitives to low-level crypto-protocols
- formal proofs of probabilistic protocols

Secure Distributed Computations and their Proofs

- Secure Implementations for Typed Session Abstractions (v1 and v2)
- Cryptographic Enforcement of Information-Flow Security
- Secure Audit Logs
- Automated Verifications of Protocol Implementations
- CertiCrypt: Formal Proofs for Computational Cryptography



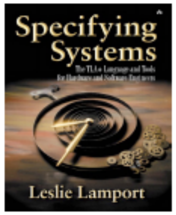
INRIA
MICROSOFT RESEARCH

Tools for formal proofs

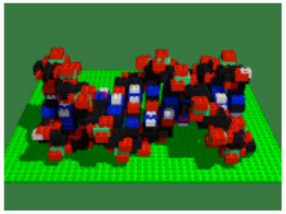
Damien Doligez, INRIA Rocq.
Kaustuv Chaudhury, MSR-INRIA (postdoc)
Leslie Lamport, MSRSV
Stephan Merz, INRIA Lorraine

Natural proofs

- first-order set theory + temporal logic
- specification/verification of concurrent programs.
- tools for automatic theorem proving



TLA+



tools for proofs

Zenon

```
EXTENDS Naturals

------------------------------------------------------------------------

(* First some general logical axioms pulled from the trusted base *)

(* The following is a specific instance of a theorem provable by the Peano axioms *)
THEOREM TwoIsNotOne ==
  2 # 1
PROOF OMITTED


THEOREM NegElim ==
  ASSUME
    NEW CONSTANT A,
    A, ~A
  PROVE
    FALSE
PROOF OMITTED

THEOREM ImplIntro ==
  ASSUME
    NEW CONSTANT A, NEW CONSTANT B,
    ASSUME A PROVE B
  PROVE A => B
PROOF OMITTED
```

```
(* The main definitions and lemmas (proofs omitted) *)

Divides(d, n) ==
  /\ d \in Nat
  /\ n \in Nat
  /\ \E q \in Nat : n = d * q

THEOREM DivLemma ==
  \A d, n \in Nat : Divides(d, n) => \E r \in Nat : n = r * d
PROOF OMITTED

Prime(x) ==
  /\ x \in Nat
  /\ \A d \in Nat : Divides(d, x) => \/ d = 1
                                     \/ d = x

PrimeNat == {x \in Nat : Prime(x)}

THEOREM TwoIsPrime == 2 \in PrimeNat
PROOF OMITTED

THEOREM SquareLemma ==
  \A p \in PrimeNat, x \in Nat :
     Divides(p, x^2) => Divides(p, x)
PROOF OMITTED
```

```
(**
 * Main theorem: there is no irreducible rational number x/y whose
 * square is 2.
 *)
THEOREM SqrtTwoIrrational ==
  \A x, y \in Nat : Coprime(x, y) => x^2 /= 2 * y^2
PROOF <1>1. ASSUME
              NEW x \in Nat,
              NEW y \in Nat,
              coprimality:: Coprime(x, y),
              main:: x^2 = 2 * y^2
            PROVE
              FALSE
            PROOF <2>1. Divides(2, x)
                  PROOF <3>1. Divides(2, x^2)
                              BY <1>1!3
                        <3>2. QED
                              BY <3>1, TwoIsPrime, SquareLemma
                  <2>2. Divides(2, y)
                        PROOF <3>1. PICK r \in Nat : x = 2 * r
                                    BY <2>1, DivLemma
                              <3>2. x^2 = 2 * (2 * r^2)
                                    BY <3>1
                              <3>3. 2 * y^2 = 2 * (2 * r^2)
                                    BY <1>1!main, <3>2
                              <3>4. y^2 = 2 * r^2
```

```
      <3>2. QED
            BY <3>1, TwoIsPrime, SquareLemma
   <2>2. Divides(2, y)
         PROOF <3>1. PICK r \in Nat : x = 2 * r
                     BY <2>1, DivLemma
               <3>2. x^2 = 2 * (2 * r^2)
                     BY <3>1
               <3>3. 2 * y^2 = 2 * (2 * r^2)
                     BY <1>1!main, <3>2
               <3>4. y^2 = 2 * r^2
                     BY <3>2, LeftCancellationLemma
               <3>5. QED
                     BY <3>3, TwoIsPrime, SquareLemma
   <2>3. ~ (Divides(2, y))
         PROOF <3>1. \A d \in Nat : (Divides(d, x) /\ Divides(d, y)) => d = 1
                     BY <1>1!coprimality
               <3>2. 2 = 1
                     BY <2>1, <2>2, <3>1
               <3>3. QED
                     BY <3>2, TwoIsNotOne
      <2>4. QED
            BY <2>2, <2>3, NegElim
<1>2. QED
   BY <1>1, ImplIntro, ForallIntro
```

Logics in track A

Math. components	Coq	higher-order + reflection
Security	PV/CV	applied pi-calculus + stochastic
Spec. / Verif.	TLA+	1st order + ZF + temporal

Track B

Computational Sciences
Scientific Information Interaction

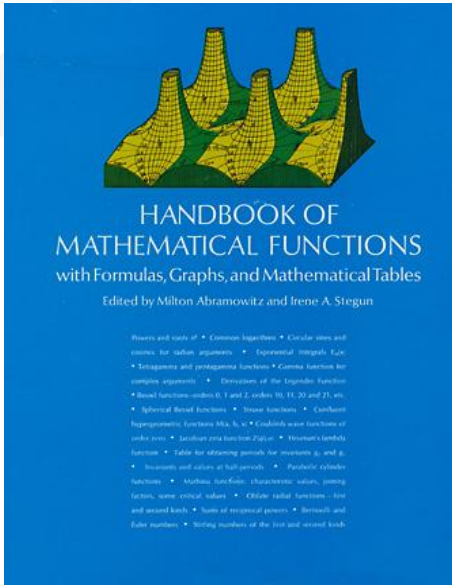
Dynamic dictionary of math functions

Bruno Salvy, INRIA Rocq.,
Alin Bostan, INRIA Rocq.,
Frédéric Chyzak, INRIA Rocq.
Henry Cohn, [Theory Group] MSRR
Alexandre Benoit, MSR-INRIA (intern)
Marc Mezzarobba, MSR-INRIA (intern)

Computer Algebra and Web for useful functions,

– dynamic tables of their properties.
– generation of programs to compute them.

Maple 11



CENTRE DE RECHERCHE COMMUN
INRIA
MICROSOFT RESEARCH

9. Bessel Functions of Integer Order

Mathematical Properties

Notation

The tables in this chapter are for Bessel functions of integer order; the text treats general orders. The conventions used are:

$z=x+iy$; x, y real.

n is a positive integer or zero.

ν, μ are unrestricted except where otherwise indicated; ν is supposed real in the sections devoted to Kelvin functions **9.9, 9.10,** and **9.11.**

The notation used for the Bessel functions is that of Watson [9.15] and the British Association and Royal Society Mathematical Tables. The function $Y_\nu(z)$ is often denoted $N_\nu(z)$ by physicists and European workers.

Other notations are those of:

Aldis, Airey:

$$G_n(z) \text{ for } -\tfrac{1}{2}\pi Y_n(z), K_n(z) \text{ for } (-)^n K_n(z).$$

Clifford:

$$C_n(x) \text{ for } x^{-\frac{1}{2}n} J_n(2\sqrt{x}).$$

Gray, Mathews and MacRobert [9.9]:

$$Y_n(z) \text{ for } \tfrac{1}{2}\pi Y_n(z)+(\ln 2-\gamma)J_n(z),$$

$$\overline{Y}_\nu(z) \text{ for } \pi e^{\nu\pi i}\sec(\nu\pi)Y_\nu(z),$$

$$G_\nu(z) \text{ for } \tfrac{1}{2}\pi i H_\nu^{(1)}(z).$$

Jahnke, Emde and Lösch [9.32]:

$$\Lambda_\nu(z) \text{ for } \Gamma(\nu+1)(\tfrac{1}{2}z)^{-\nu}J_\nu(z).$$

Jeffreys:

$$Hs_\nu(z) \text{ for } H_\nu^{(1)}(z),\ Hi_\nu(z) \text{ for } H_\nu^{(2)}(z),$$

$$Kh_\nu(z) \text{ for } (2/\pi)K_\nu(z).$$

Heine:

$$K_n(z) \text{ for} -\tfrac{1}{2}\pi Y_n(z).$$

Neumann:

$$Y^n(z) \text{ for } \tfrac{1}{2}\pi Y_n(z)+(\ln 2-\gamma)J_n(z).$$

Whittaker and Watson [9.18]:

$$K_\nu(z) \text{ for } \cos(\nu\pi)K_\nu(z).$$

Bessel Functions J and Y

9.1. Definitions and Elementary Properties

Differential Equation

9.1.1
$$z^2\frac{d^2w}{dz^2}+z\frac{dw}{dz}+(z^2-\nu^2)w=0$$

Solutions are the Bessel functions of the first kind $J_{\pm\nu}(z)$, of the second kind $Y_\nu(z)$ (also called Weber's function) and of the third kind $H_\nu^{(1)}(z)$, $H_\nu^{(2)}(z)$ (also called the Hankel functions). Each is a regular (holomorphic) function of z throughout the z-plane cut along the negative real axis, and for fixed $z(\neq 0)$ each is an entire (integral) function of ν. When $\nu=\pm n$, $J_\nu(z)$ has no branch point and is an entire (integral) function of z.

Important features of the various solutions are as follows: $J_\nu(z)(\mathscr{R}\nu\geq 0)$ is bounded as $z\to 0$ in any bounded range of arg z. $J_\nu(z)$ and $J_{-\nu}(z)$ are linearly independent except when ν is an integer. $J_\nu(z)$ and $Y_\nu(z)$ are linearly independent for all values of ν.

$H_\nu^{(1)}(z)$ tends to zero as $|z|\to\infty$ in the sector $0<\arg z<\pi$; $H_\nu^{(2)}(z)$ tends to zero as $|z|\to\infty$ in the sector $-\pi<\arg z<0$. For all values of ν, $H_\nu^{(1)}(z)$ and $H_\nu^{(2)}(z)$ are linearly independent.

Relations Between Solutions

9.1.2
$$Y_\nu(z)=\frac{J_\nu(z)\cos(\nu\pi)-J_{-\nu}(z)}{\sin(\nu\pi)}$$

The right of this equation is replaced by its limiting value if ν is an integer or zero.

9.1.3
$$H_\nu^{(1)}(z)=J_\nu(z)+iY_\nu(z)$$
$$=i\csc(\nu\pi)\{e^{-\nu\pi i}J_\nu(z)-J_{-\nu}(z)\}$$

9.1.4
$$H_\nu^{(2)}(z)=J_\nu(z)-iY_\nu(z)$$
$$=i\csc(\nu\pi)\{J_{-\nu}(z)-e^{\nu\pi i}J_\nu(z)\}$$

9.1.5 $\quad J_{-n}(z)=(-)^nJ_n(z) \qquad Y_{-n}(z)=(-)^nY_n(z)$

9.1.6 $\quad H_{-\nu}^{(1)}(z)=e^{\nu\pi i}H_\nu^{(1)}(z) \qquad H_{-\nu}^{(2)}(z)=e^{-\nu\pi i}H_\nu^{(2)}(z)$

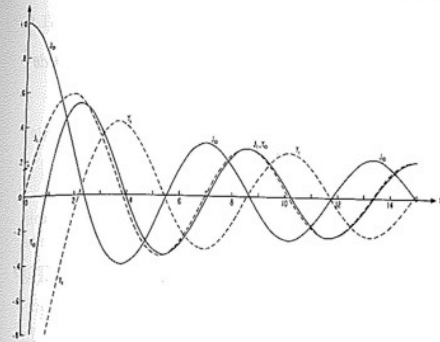


Figure 9.1. $J_0(x)$, $Y_0(x)$, $J_1(x)$, $Y_1(x)$.

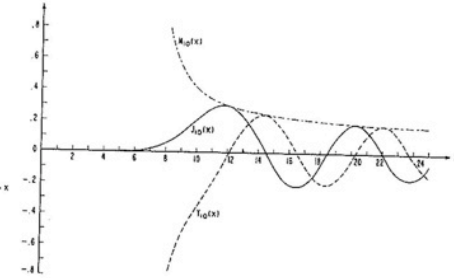


Figure 9.2. $J_{10}(x)$, $Y_{10}(x)$, *and* $M_{10}(x)=\sqrt{J_{10}^2(x)+Y_{10}^2(x)}$.

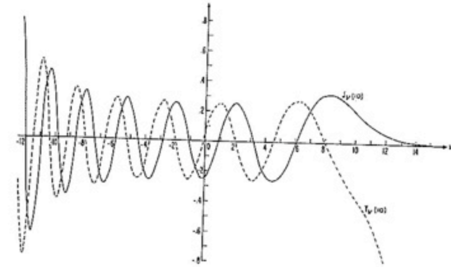


Figure 9.3. $J_\nu(10)$ *and* $Y_\nu(10)$.

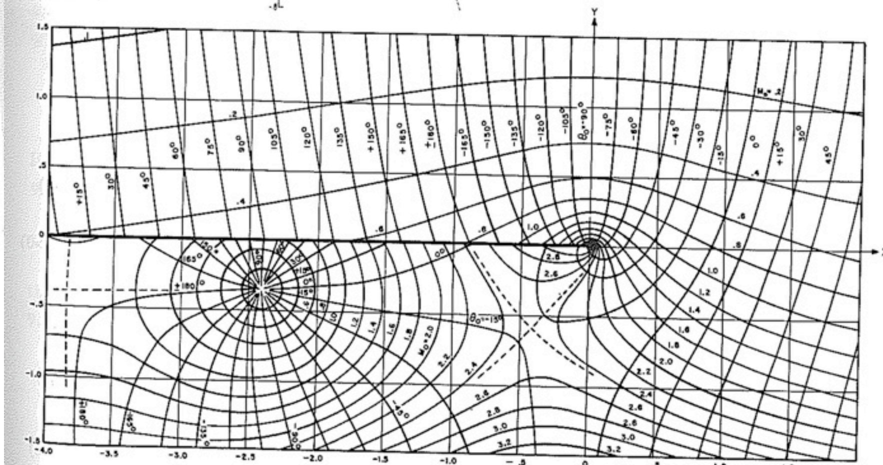


Figure 9.4. *Contour lines of the modulus and phase of the Hankel Function* $H_0^{(1)}(x+iy)=M_0e^{i\theta_0}$. From E. Jahnke, F. Emde, and F. Lösch, Tables of higher functions, McGraw-Hill Book Co., Inc., New York, N.Y., 1960 (with permission).

Limiting Forms for Small Arguments

When ν is fixed and $z\to 0$

9.1.7 $J_\nu(z)\sim(\tfrac12 z)^\nu/\Gamma(\nu+1) \qquad (\nu\neq-1,-2,-3,\ldots)$

9.1.8 $Y_0(z)\sim -iH_0^{(1)}(z)\sim iH_0^{(2)}(z)\sim(2/\pi)\ln z$

9.1.9

$$Y_\nu(z)\sim -iH_\nu^{(1)}(z)\sim iH_\nu^{(2)}(z)\sim -(1/\pi)\Gamma(\nu)(\tfrac12 z)^{-\nu} \qquad (\mathscr{R}\nu>0)$$

Ascending Series

9.1.10 $$J_\nu(z)=(\tfrac12 z)^\nu\sum_{k=0}^\infty \frac{(-\frac14 z^2)^k}{k!\,\Gamma(\nu+k+1)}$$

9.1.11

$$Y_n(z)=-\frac{(\frac12 z)^{-n}}{\pi}\sum_{k=0}^{n-1}\frac{(n-k-1)!}{k!}(\tfrac14 z^2)^k+\frac{2}{\pi}\ln(\tfrac12 z)J_n(z)-\frac{(\frac12 z)^n}{\pi}\sum_{k=0}^\infty\{\psi(k+1)+\psi(n+k+1)\}\frac{(-\frac14 z^2)^k}{k!\,(n+k)!}$$

where $\psi(n)$ is given by **6.3.2.**

9.1.12 $$J_0(z)=1-\frac{\frac14 z^2}{(1!)^2}+\frac{(\frac14 z^2)^2}{(2!)^2}-\frac{(\frac14 z^2)^3}{(3!)^2}+\ldots$$

9.1.13

$$Y_0(z)=\frac{2}{\pi}\{\ln(\tfrac12 z)+\gamma\}J_0(z)+\frac{2}{\pi}\Big\{\frac{\frac14 z^2}{(1!)^2}-(1+\tfrac12)\frac{(\frac14 z^2)^2}{(2!)^2}+(1+\tfrac12+\tfrac13)\frac{(\frac14 z^2)^3}{(3!)^2}-\ldots\Big\}$$

9.1.14

$$J_\nu(z)J_\mu(z)=(\tfrac12 z)^{\nu+\mu}\sum_{k=0}^\infty\frac{(-)^k\Gamma(\nu+\mu+2k+1)(\frac14 z^2)^k}{\Gamma(\nu+k+1)\Gamma(\mu+k+1)\Gamma(\nu+\mu+k+1)\,k!}$$

Wronskians

9.1.15

$$W\{J_\nu(z),J_{-\nu}(z)\}=J_{\nu+1}(z)J_{-\nu}(z)+J_\nu(z)J_{-(\nu+1)}(z)=-2\sin(\nu\pi)/(\pi z)$$

9.1.16

$$W\{J_\nu(z),Y_\nu(z)\}=J_{\nu+1}(z)Y_\nu(z)-J_\nu(z)Y_{\nu+1}(z)=2/(\pi z)$$

9.1.17

$$W\{H_\nu^{(1)}(z),H_\nu^{(2)}(z)\}=H_{\nu+1}^{(1)}(z)H_\nu^{(2)}(z)-H_\nu^{(1)}(z)H_{\nu+1}^{(2)}(z)=-4i/(\pi z)$$

Integral Representations

9.1.18 $$J_0(z)=\frac1\pi\int_0^\pi\cos(z\sin\theta)d\theta=\frac1\pi\int_0^\pi\cos(z\cos\theta)d\theta$$

9.1.19 $$Y_0(z)=\frac{4}{\pi^2}\int_0^{\frac12\pi}\cos(z\cos\theta)\{\gamma+\ln(2z\sin^2\theta)\}\,d\theta$$

9.1.20

$$J_\nu(z)=\frac{(\frac12 z)^\nu}{\pi^{\frac12}\Gamma(\nu+\frac12)}\int_0^\pi\cos(z\cos\theta)\sin^{2\nu}\theta\,d\theta=\frac{2(\frac12 z)^\nu}{\pi^{\frac12}\Gamma(\nu+\frac12)}\int_0^1(1-t^2)^{\nu-\frac12}\cos(zt)dt \quad (\mathscr{R}\nu>-\tfrac12)$$

9.1.21

$$J_n(z)=\frac1\pi\int_0^\pi\cos(z\sin\theta-n\theta)\,d\theta=\frac{i^{-n}}{\pi}\int_0^\pi e^{iz\cos\theta}\cos(n\theta)d\theta$$

9.1.22

$$J_\nu(z)=\frac1\pi\int_0^\pi\cos(z\sin\theta-\nu\theta)d\theta-\frac{\sin(\nu\pi)}{\pi}\int_0^\infty e^{-z\sinh t-\nu t}dt \quad (|\arg z|<\tfrac12\pi)$$

$$Y_\nu(z)=\frac1\pi\int_0^\pi\sin(z\sin\theta-\nu\theta)d\theta-\frac1\pi\int_0^\infty\{e^{\nu t}+e^{-\nu t}\cos(\nu\pi)\}e^{-z\sinh t}dt \quad (|\arg z|<\tfrac12\pi)$$

9.1.23

$$J_0(x)=\frac2\pi\int_0^\infty\sin(x\cosh t)dt \quad (x>0)$$

$$Y_0(x)=-\frac2\pi\int_0^\infty\cos(x\cosh t)dt \quad (x>0)$$

9.1.24

$$J_\nu(x)=\frac{2(\frac12 x)^{-\nu}}{\pi^{\frac12}\Gamma(\frac12-\nu)}\int_1^\infty\frac{\sin(xt)\,dt}{(t^2-1)^{\nu+\frac12}} \quad (|\mathscr{R}\nu|<\tfrac12, x>0)$$

$$Y_\nu(x)=-\frac{2(\frac12 x)^{-\nu}}{\pi^{\frac12}\Gamma(\frac12-\nu)}\int_1^\infty\frac{\cos(xt)\,dt}{(t^2-1)^{\nu+\frac12}} \quad (|\mathscr{R}\nu|<\tfrac12, x>0)$$

9.1.25

$$H_\nu^{(1)}(z)=\frac{1}{\pi i}\int_{-\infty}^{\infty+\pi i}e^{z\sinh t-\nu t}dt \quad (|\arg z|<\tfrac12\pi)$$

$$H_\nu^{(2)}(z)=-\frac{1}{\pi i}\int_{-\infty}^{\infty-\pi i}e^{z\sinh t-\nu t}dt \quad (|\arg z|<\tfrac12\pi)$$

9.1.26

$$J_\nu(x)=\frac{1}{2\pi i}\int_{-i\infty}^{i\infty}\frac{\Gamma(-t)(\frac12 x)^{\nu+2t}}{\Gamma(\nu+t+1)}dt \quad (\mathscr{R}\nu>0, x>0)$$

In the last integral the path of integration must lie to the left of the points $t=0, 1, 2, \ldots$.

Recurrence Relations

9.1.27

$$\mathscr{C}_{\nu-1}(z)+\mathscr{C}_{\nu+1}(z)=\frac{2\nu}{z}\mathscr{C}_\nu(z)$$

$$\mathscr{C}_{\nu-1}(z)-\mathscr{C}_{\nu+1}(z)=2\mathscr{C}_\nu'(z)$$

$$\mathscr{C}_\nu'(z)=\mathscr{C}_{\nu-1}(z)-\frac{\nu}{z}\mathscr{C}_\nu(z)$$

$$\mathscr{C}_\nu'(z)=-\mathscr{C}_{\nu+1}(z)+\frac{\nu}{z}\mathscr{C}_\nu(z)$$

$\mathscr{C}$ denotes $J, Y, H^{(1)}, H^{(2)}$ or any linear combination of these functions, the coefficients in which are independent of z and ν.

9.1.28 $J_0'(z)=-J_1(z) \qquad Y_0'(z)=-Y_1(z)$

If $f_\nu(z)=z^p\mathscr{C}_\nu(\lambda z^q)$ where p, q, λ are independent of ν, then

9.1.29

$$f_{\nu-1}(z)+f_{\nu+1}(z)=(2\nu/\lambda)z^{-q}f_\nu(z)$$

$$(p+\nu q)f_{\nu-1}(z)+(p-\nu q)f_{\nu+1}(z)=(2\nu/\lambda)z^{1-q}f_\nu'(z)$$

$$zf_\nu'(z)=\lambda qz^qf_{\nu-1}(z)+(p-\nu q)f_\nu(z)$$

$$zf_\nu'(z)=-\lambda qz^qf_{\nu+1}(z)+(p+\nu q)f_\nu(z)$$

Formulas for Derivatives

9.1.30

$$\left(\frac1z\frac{d}{dz}\right)^k\{z^\nu\mathscr{C}_\nu(z)\}=z^{\nu-k}\mathscr{C}_{\nu-k}(z)$$

$$\left(\frac1z\frac{d}{dz}\right)^k\{z^{-\nu}\mathscr{C}_\nu(z)\}=(-)^kz^{-\nu-k}\mathscr{C}_{\nu+k}(z) \qquad (k=0,1,2,\ldots)$$

9.1.31

$$\mathscr{C}_\nu^{(k)}(z)=\frac{1}{2^k}\left\{\mathscr{C}_{\nu-k}(z)-\binom{k}{1}\mathscr{C}_{\nu-k+2}(z)+\binom{k}{2}\mathscr{C}_{\nu-k+4}(z)-\ldots+(-)^k\mathscr{C}_{\nu+k}(z)\right\} \qquad (k=0,1,2,\ldots)$$

Recurrence Relations for Cross-Products

If

9.1.32

$$p_\nu=J_\nu(a)Y_\nu(b)-J_\nu(b)Y_\nu(a)$$

$$q_\nu=J_\nu(a)Y_\nu'(b)-J_\nu'(b)Y_\nu(a)$$

$$r_\nu=J_\nu'(a)Y_\nu(b)-J_\nu(b)Y_\nu'(a)$$

$$s_\nu=J_\nu'(a)Y_\nu'(b)-J_\nu'(b)Y_\nu'(a)$$

then

9.1.33

$$p_{\nu+1}-p_{\nu-1}=-\frac{2\nu}{a}q_\nu-\frac{2\nu}{b}r_\nu$$

$$q_{\nu+1}+r_\nu=\frac{\nu}{a}p_\nu-\frac{\nu+1}{b}p_{\nu+1}$$

$$r_{\nu+1}+q_\nu=\frac{\nu}{b}p_\nu-\frac{\nu+1}{a}p_{\nu+1}$$

$$s_\nu=\frac12 p_{\nu+1}+\frac12 p_{\nu-1}-\frac{\nu^2}{ab}p_\nu$$

and

9.1.34 $$p_\nu s_\nu-q_\nu r_\nu=\frac{4}{\pi^2 ab}$$

Analytic Continuation

In **9.1.35** to **9.1.38**, m is an integer.

9.1.35 $$J_\nu(ze^{m\pi i})=e^{m\nu\pi i}J_\nu(z)$$

9.1.36

$$Y_\nu(ze^{m\pi i})=e^{-m\nu\pi i}Y_\nu(z)+2i\sin(m\nu\pi)\cot(\nu\pi)J_\nu(z)$$

9.1.37

$$\sin(\nu\pi)H_\nu^{(1)}(ze^{m\pi i})=-\sin\{(m-1)\nu\pi\}H_\nu^{(1)}(z)-e^{-\nu\pi i}\sin(m\nu\pi)H_\nu^{(2)}(z)$$

9.1.38

$$\sin(\nu\pi)H_\nu^{(2)}(ze^{m\pi i})=\sin\{(m+1)\nu\pi\}H_\nu^{(2)}(z)+e^{\nu\pi i}\sin(m\nu\pi)H_\nu^{(1)}(z)$$

9.1.39

$$H_\nu^{(1)}(ze^{\pi i})=-e^{-\nu\pi i}H_\nu^{(2)}(z)$$

$$H_\nu^{(2)}(ze^{-\pi i})=-e^{\nu\pi i}H_\nu^{(1)}(z)$$

9.1.40

$$J_\nu(\bar z)=\overline{J_\nu(z)} \qquad Y_\nu(\bar z)=\overline{Y_\nu(z)}$$

$$H_\nu^{(1)}(\bar z)=\overline{H_\nu^{(2)}(z)} \qquad H_\nu^{(2)}(\bar z)=\overline{H_\nu^{(1)}(z)} \qquad (\nu \text{ real})$$

Generating Function and Associated Series

9.1.41 $$e^{\frac12 z(t-1/t)}=\sum_{k=-\infty}^{\infty}t^kJ_k(z) \qquad (t\neq 0)$$

9.1.42 $$\cos(z\sin\theta)=J_0(z)+2\sum_{k=1}^\infty J_{2k}(z)\cos(2k\theta)$$

9.1.43 $$\sin(z\sin\theta)=2\sum_{k=0}^\infty J_{2k+1}(z)\sin\{(2k+1)\theta\}$$

9.1.44

$$\cos(z\cos\theta)=J_0(z)+2\sum_{k=1}^\infty(-)^kJ_{2k}(z)\cos(2k\theta)$$

9.1.45

$$\sin(z\cos\theta)=2\sum_{k=0}^\infty(-)^kJ_{2k+1}(z)\cos\{(2k+1)\theta\}$$

9.1.46 $$1=J_0(z)+2J_2(z)+2J_4(z)+2J_6(z)+\ldots$$

9.1.47

$$\cos z=J_0(z)-2J_2(z)+2J_4(z)-2J_6(z)+\ldots$$

9.1.48 $$\sin z=2J_1(z)-2J_3(z)+2J_5(z)-\ldots$$

9.1.72

$$\lim \left\{\nu^{\mu} Q_{\nu}^{-\mu}\left(\cos \frac{x}{\nu}\right)\right\}=-\tfrac{1}{2}\pi Y_{\mu}(x) \qquad (x>0)$$

For $P_{\nu}^{-\mu}$ and $Q_{\nu}^{-\mu}$, see chapter 8.

Continued Fractions

9.1.73

$$\frac{J_{\nu}(z)}{J_{\nu-1}(z)}=\frac{1}{2\nu z^{-1}-}\,\frac{1}{2(\nu+1)z^{-1}-}\,\frac{1}{2(\nu+2)z^{-1}-}\cdots$$
$$=\frac{\frac{1}{2}z/\nu}{1-}\,\frac{\frac{1}{4}z^2/\{\nu(\nu+1)\}}{1-}\,\frac{\frac{1}{4}z^2/\{(\nu+1)(\nu+2)\}}{1-}\cdots$$

Multiplication Theorem

9.1.74

$$\mathscr{C}_{\nu}(\lambda z)=\lambda^{\pm\nu}\sum_{k=0}^{\infty}\frac{(\mp)^k(\lambda^2-1)^k(\frac{1}{2}z)^k}{k!}\mathscr{C}_{\nu\pm k}(z) \qquad (|\lambda^2-1|<1)$$

If $\mathscr{C}=J$ and the upper signs are taken, the restriction on λ is unnecessary.

This theorem will furnish expansions of $\mathscr{C}_{\nu}(re^{i\theta})$ in terms of $\mathscr{C}_{\nu\pm k}(r)$.

Addition Theorems

Neumann's

9.1.75 $$\mathscr{C}_{\nu}(u\pm v)=\sum_{k=-\infty}^{\infty}\mathscr{C}_{\nu\mp k}(u)J_k(v) \qquad (|v|<|u|)$$

The restriction $|v|<|u|$ is unnecessary when $\mathscr{C}=J$ and ν is an integer or zero. Special cases are

9.1.76 $$1=J_0^2(z)+2\sum_{k=1}^{\infty}J_k^2(z)$$

9.1.77

$$0=\sum_{k=0}^{2n}(-)^kJ_k(z)J_{2n-k}(z)+2\sum_{k=1}^{\infty}J_k(z)J_{2n+k}(z) \quad (n\geq 1)$$

9.1.78

$$J_n(2z)=\sum_{k=0}^{n}J_k(z)J_{n-k}(z)+2\sum_{k=1}^{\infty}(-)^kJ_k(z)J_{n+k}(z)$$

Graf's

9.1.79

$$\mathscr{C}_{\nu}(w)\begin{matrix}\cos\\ \sin\end{matrix}\nu\chi=\sum_{k=-\infty}^{\infty}\mathscr{C}_{\nu+k}(u)J_k(v)\begin{matrix}\cos\\ \sin\end{matrix}k\alpha \quad (|ve^{\pm i\alpha}|<|u|)$$

Gegenbauer's

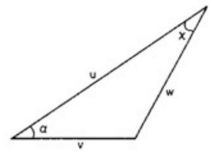
9.1.80

$$\frac{\mathscr{C}_{\nu}(w)}{w^{\nu}}=2^{\nu}\Gamma(\nu)\sum_{k=0}^{\infty}(\nu+k)\frac{\mathscr{C}_{\nu+k}(u)}{u^{\nu}}\frac{J_{\nu+k}(v)}{v^{\nu}}C_k^{(\nu)}(\cos\alpha)$$
$$(\nu\neq 0,-1,\ldots,|ve^{\pm i\alpha}|<|u|)$$

In **9.1.79** and **9.1.80**,

$$w=\sqrt{(u^2+v^2-2uv\cos\alpha)},$$
$$u-v\cos\alpha=w\cos\chi,\quad v\sin\alpha=w\sin\chi$$

the branches being chosen so that $w\to u$ and $\chi\to 0$ as $v\to 0$. $C_k^{(\nu)}(\cos\alpha)$ is Gegenbauer's polynomial (see chapter 22).



Gegenbauer's addition theorem.

If u, v are real and positive and $0\leq\alpha\leq\pi$, then w, χ are real and non-negative, and the geometrical relationship of the variables is shown in the diagram.

The restrictions $|ve^{\pm i\alpha}|<|u|$ are unnecessary in **9.1.79** when $\mathscr{C}=J$ and ν is an integer or zero, and in **9.1.80** when $\mathscr{C}=J$.

Degenerate Form ($u=\infty$):

9.1.81

$$e^{iv\cos\alpha}=\Gamma(\nu)(\tfrac{1}{2}v)^{-\nu}\sum_{k=0}^{\infty}(\nu+k)i^kJ_{\nu+k}(v)C_k^{(\nu)}(\cos\alpha)$$
$$(\nu\neq 0,-1,\ldots)$$

Neumann's Expansion of an Arbitrary Function in a Series of Bessel Functions

9.1.82 $$f(z)=a_0J_0(z)+2\sum_{k=1}^{\infty}a_kJ_k(z) \qquad (|z|<c)$$

where c is the distance of the nearest singularity of $f(z)$ from $z=0$,

9.1.83 $$a_k=\frac{1}{2\pi i}\int_{|t|=c'}f(t)O_k(t)dt \qquad (0<c'<c)$$

and $O_k(t)$ is Neumann's polynomial. The latter is defined by the generating function

9.1.84

$$\frac{1}{t-z}=J_0(z)O_0(t)+2\sum_{k=1}^{\infty}J_k(z)O_k(t) \qquad (|z|<|t|)$$

$O_n(t)$ is a polynomial of degree $n+1$ in $1/t$; $O_0(t)=1/t$,

9.1.85

$$O_n(t)=\frac{1}{4}\sum_{k=0}^{\leq\frac{1}{2}n}\frac{n(n-k-1)!}{k!}\left(\frac{2}{t}\right)^{n-2k+1} \qquad (n=1,2,\ldots)$$

The more general form of expansion

9.1.86 $$f(z)=a_0J_{\nu}(z)+2\sum_{k=1}^{\infty}a_kJ_{\nu+k}(z)$$

	$f(s)$			$F(t)$
29.3.126	$e^{as}E_1(as)$	$(a>0)$	5	$\frac{1}{t+a}$
29.3.127	$\frac{1}{a}-se^{as}E_1(as)$	$(a>0)$	5	$\frac{1}{(t+a)^2}$
29.3.128	$a^{1-n}e^{as}E_n(as)$	$(a>0; n=0,1,2,\ldots)$	5	$\frac{1}{(t+a)^n}$
29.3.129	$\left[\frac{\pi}{2}-\mathrm{Si}(s)\right]\cos s+\mathrm{Ci}(s)\sin s$		5	$\frac{1}{t^2+1}$

29.4. Table of Laplace-Stieltjes Transforms [4]

	$\phi(s)$		$\Phi(t)$
29.4.1	$\int_0^{\infty}e^{-st}d\Phi(t)$		$\Phi(t)$
29.4.2	e^{-ks}	$(k>0)$	$u(t-k)$
29.4.3	$\frac{1}{1-e^{-ks}}$	$(k>0)$	$\sum_{n=0}^{\infty}u(t-nk)$
29.4.4	$\frac{1}{1+e^{-ks}}$	$(k>0)$	$\sum_{n=0}^{\infty}(-1)^nu(t-nk)$
29.4.5	$\frac{1}{\sinh ks}$	$(k>0)$	$2\sum_{n=0}^{\infty}u[t-(2n+1)k]$
29.4.6	$\frac{1}{\cosh ks}$	$(k>0)$	$2\sum_{n=0}^{\infty}(-1)^nu[t-(2n+1)k]$
29.4.7	$\tanh ks$	$(k>0)$	$u(t)+2\sum_{n=1}^{\infty}(-1)^nu(t-2nk)$
29.4.8	$\frac{1}{\sinh(ks+a)}$	$(k>0)$	$2\sum_{n=0}^{\infty}e^{-(2n+1)a}u[t-(2n+1)k]$
29.4.9	$\frac{e^{-hs}}{\sinh(ks+a)}$	$(k>0, h>0)$	$2\sum_{n=0}^{\infty}e^{-(2n+1)a}u[t-h-(2n+1)k]$
29.4.10	$\frac{\sinh(hs+b)}{\sinh(ks+a)}$	$(0<h<k)$	$\sum_{n=0}^{\infty}e^{-(2n+1)a}\{e^bu[t+h-(2n+1)k]$ $-e^{-b}u[t-h-(2n+1)k]\}$
29.4.11	$\sum_{n=0}^{\infty}a_ne^{-k_ns}$	$(0<k_0<k_1<\ldots)$	$\sum_{n=0}^{\infty}a_nu(t-k_n)$

For the definition of the Laplace-Stieltjes transform see [29.7]. In practice, Laplace-Stieltjes transforms are often written as ordinary Laplace transforms involving Dirac's delta function $\delta(t)$. This "function" may formally be considered as the derivative of the unit step function, $du(t)=\delta(t)dt$, so that $\int_{-\infty}^{x}du(t)=\int_{-\infty}^{x}\delta(t)dt=\begin{cases}0 & (x<0)\\ 1 & (x>0).\end{cases}$ The correspondence **29.4.2**, for instance, then assumes the form $e^{-ks}=\int_0^{\infty}e^{-st}\delta(t-k)dt$.

[4] Adapted by permission from P. M. Morse and H. Feshbach, Methods of theoretical physics, vols. 1, 2, McGraw-Hill Book Co., Inc., New York, N.Y., 1953.

9. Bessel Functions of Integer Order

Mathematical Properties

Notation

The tables in this chapter are for Bessel functions of integer order; the text treats general orders. The conventions used are:

$z = x + iy$; x, y real.

n is a positive integer or zero.

ν, μ are unrestricted except where otherwise indicated; ν is supposed real in the sections devoted to Kelvin functions **9.9**, **9.10**, and **9.11**.

The notation used for the Bessel functions is that of Watson [9.15] and the British Association and Royal Society Mathematical Tables. The function $Y_\nu(z)$ is often denoted $N_\nu(z)$ by physicists and European workers.

Other notations are those of:

Aldis, Airey:

$$G_n(z) \text{ for } -\tfrac{1}{2}\pi Y_n(z), K_n(z) \text{ for } (-)^n K_n(z).$$

Clifford:

$$C_n(x) \text{ for } x^{-\frac{1}{2}n} J_n(2\sqrt{x}).$$

Gray, Mathews and MacRobert [9.9]:

$$Y_n(z) \text{ for } \tfrac{1}{2}\pi Y_n(z) + (\ln 2 - \gamma) J_n(z),$$

$$\overline{Y}_\nu(z) \text{ for } \pi e^{\nu\pi i} \sec(\nu\pi) Y_\nu(z),$$

$$G_\nu(z) \text{ for } \tfrac{1}{2}\pi i H_\nu^{(1)}(z).$$

Jahnke, Emde and Lösch [9.32]:

$$\Lambda_\nu(z) \text{ for } \Gamma(\nu+1)(\tfrac{1}{2}z)^{-\nu} J_\nu(z).$$

Jeffreys:

$Hs_\nu(z)$ for $H_\nu^{(1)}(z)$, $Hi_\nu(z)$ for $H_\nu^{(2)}(z)$,

$Kh_\nu(z)$ for $(2/\pi) K_\nu(z)$.

Heine:

$$K_n(z) \text{ for } -\tfrac{1}{2}\pi Y_n(z).$$

Neumann:

$$Y^n(z) \text{ for } \tfrac{1}{2}\pi Y_n(z) + (\ln 2 - \gamma) J_n(z).$$

Whittaker and Watson [9.18]:

$$K_\nu(z) \text{ for } \cos(\nu\pi) K_\nu(z).$$

Bessel Functions J and Y

9.1. Definitions and Elementary Properties

Differential Equation

9.1.1
$$z^2 \frac{d^2 w}{dz^2} + z \frac{dw}{dz} + (z^2 - \nu^2) w = 0$$

Solutions are the Bessel functions of the first kind $J_{\pm\nu}(z)$, of the second kind $Y_\nu(z)$ (also called Weber's function) and of the third kind $H_\nu^{(1)}(z)$, $H_\nu^{(2)}(z)$ (also called the Hankel functions). Each is a regular (holomorphic) function of z throughout the z-plane cut along the negative real axis, and for fixed $z (\neq 0)$ each is an entire (integral) function of ν. When $\nu = \pm n$, $J_\nu(z)$ has no branch point and is an entire (integral) function of z.

Important features of the various solutions are as follows: $J_\nu(z) (\mathscr{R}\nu \geq 0)$ is bounded as $z \to 0$ in any bounded range of arg z. $J_\nu(z)$ and $J_{-\nu}(z)$ are linearly independent except when ν is an integer. $J_\nu(z)$ and $Y_\nu(z)$ are linearly independent for all values of ν.

$H_\nu^{(1)}(z)$ tends to zero as $|z| \to \infty$ in the sector $0 < \arg z < \pi$; $H_\nu^{(2)}(z)$ tends to zero as $|z| \to \infty$ in the sector $-\pi < \arg z < 0$. For all values of ν, $H_\nu^{(1)}(z)$ and $H_\nu^{(2)}(z)$ are linearly independent.

Relations Between Solutions

9.1.2
$$Y_\nu(z) = \frac{J_\nu(z)\cos(\nu\pi) - J_{-\nu}(z)}{\sin(\nu\pi)}$$

The right of this equation is replaced by its limiting value if ν is an integer or zero.

9.1.3

$$H_\nu^{(1)}(z) = J_\nu(z) + iY_\nu(z)$$
$$= i \csc(\nu\pi) \{ e^{-\nu\pi i} J_\nu(z) - J_{-\nu}(z) \}$$

9.1.4

$$H_\nu^{(2)}(z) = J_\nu(z) - iY_\nu(z)$$
$$= i \csc(\nu\pi) \{ J_{-\nu}(z) - e^{\nu\pi i} J_\nu(z) \}$$

9.1.5 $\quad J_{-n}(z) = (-)^n J_n(z) \qquad Y_{-n}(z) = (-)^n Y_n(z)$

9.1.6 $\quad H_{-\nu}^{(1)}(z) = e^{\nu\pi i} H_\nu^{(1)}(z) \qquad H_{-\nu}^{(2)}(z) = e^{-\nu\pi i} H_\nu^{(2)}(z)$

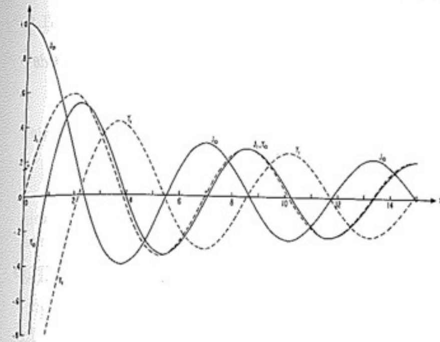


FIGURE 9.1. $J_0(x)$, $Y_0(x)$, $J_1(x)$, $Y_1(x)$.

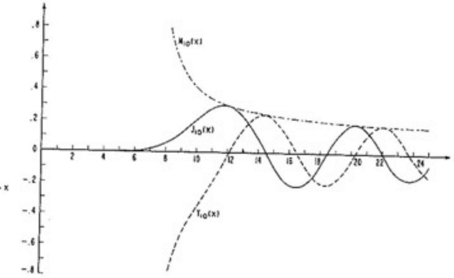


FIGURE 9.2. $J_{10}(x)$, $Y_{10}(x)$, *and* $M_{10}(x) = \sqrt{J_{10}^2(x) + Y_{10}^2(x)}$.

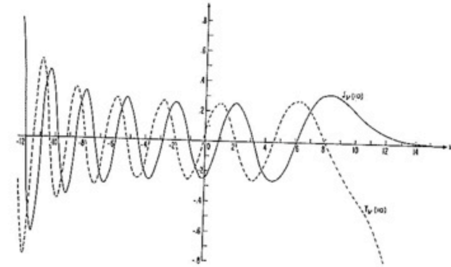


FIGURE 9.3. $J_\nu(10)$ *and* $Y_\nu(10)$.

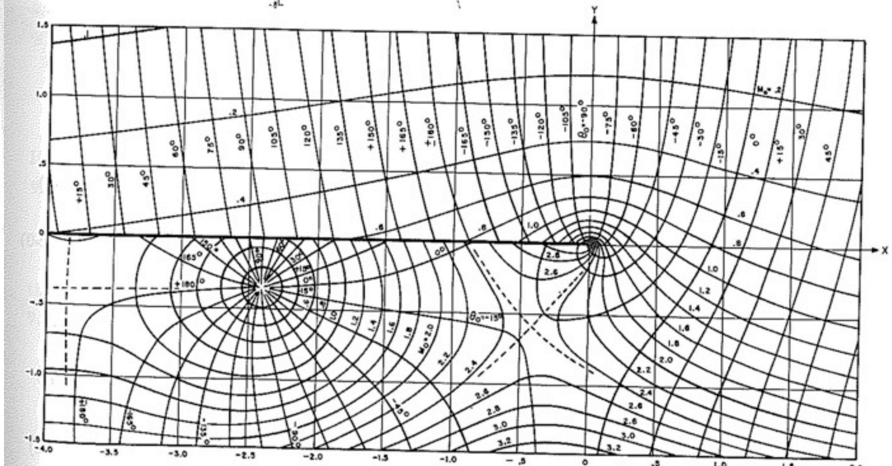


FIGURE 9.4. *Contour lines of the modulus and phase of the Hankel Function* $H_0^{(1)}(x+iy) = M_0 e^{i\theta_0}$. From E. Jahnke, F. Emde, and F. Lösch, Tables of higher functions, McGraw-Hill Book Co., Inc., New York, N.Y., 1960 (with permission).

9. Bessel Functions of Integer Order

Mathematical Properties

is chapter are for Bessel func-
rder; the text treats general
entions used are:

al.
teger or zero.
icted except where otherwise
osed real in the sections devoted
s **9.9, 9.10,** and **9.11.**
ed for the Bessel functions is
15] and the British Association
y Mathematical Tables. The
ten denoted $N_\nu(z)$ by physicists
kers.
are those of:

Bessel Functions J and Y

9.1. Definitions and Elementary Properties

Differential Equation

9.1.1 $$z^2\frac{d^2w}{dz^2}+z\frac{dw}{dz}+(z^2-\nu^2)w=0$$

Solutions are the Bessel functions of the first kind $J_{\pm\nu}(z)$, of the second kind $Y_\nu(z)$ (also called Weber's function) and of the third kind $H_\nu^{(1)}(z)$, $H_\nu^{(2)}(z)$ (also called the Hankel functions). Each is a regular (holomorphic) function of z throughout the z-plane cut along the negative real axis, and for fixed $z(\neq 0)$ each is an entire (integral) function of ν. When $\nu=\pm n$, $J_\nu(z)$ has no branch point and is an entire (integral) function of z.

Dynamic dictionary of math functions

Computer algebra:

- **classic**: polynomial to represent their roots + following tools: euclidian division, Euclid algorithm, Gröbner bases.
- **modern**: linear differential equation as data structures to represent their solutions [SaZi94, ChSa98, Chyzak00, MeSa03, Salvy05] with same tools as classical case but non-commutative.
- **prototype** ESF at http://algo.inria.fr/esf (65% of Abramowitz-Stegun)
- **todo**: interactivity, integral transforms, parametric integrals.

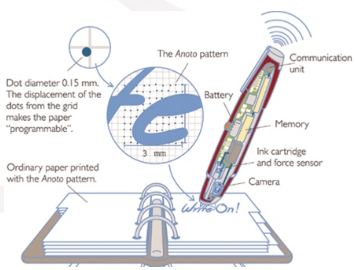
ReActivity

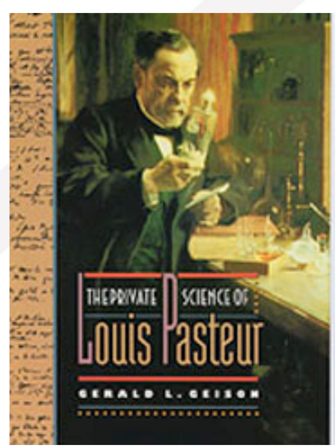
Wendy Mackay, INRIA Saclay,
J.-D. Fekete, INRIA Saclay,
Mary Czerwinski, MSRR,
George Robertson, MSRR

Michel Beaudouin-Lafon, Paris 11,
Olivier Chapuis, CNRS,
Pierre Dragicevic, INRIA Saclay,
Emmanuel Pietriga, INRIA Saclay,
Aurélien Tabard, Paris 11 (PhD)

Logs of experiments for biologists, historians, other scientists

- mixed inputs from lab notebooks and computers,
- interactive visualization of scientific activity,
- support for managing scientific workflow.





CENTRE DE RECHERCHE COMMUN
INRIA MICROSOFT RESEARCH

ReActivity

Programme:

- Log platform and infrastructure for data collection and aggregation
  - common format & share experiences,
  - apply our own visualisation tools to the logged data
- Visualisation and instrumentation of scientific data logs,
  - Visualisation of scaled to month-long or longer logs,
  - strategies of interaction and navigation for meaningful sampling of data
- Mining of desktop data and interactions with visualised activities
  - Design highly interactive tools for scientists to understand and interact with their past activies
  - Create high-level interactive reflexive views that can be manipulated and reused)

Update:

- interactive wall and collaborative workflow

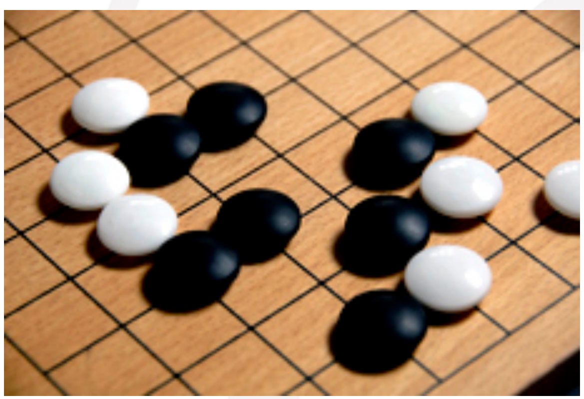
Adaptive Combinatorial Search for E-science

Youssef Hamadi, MSRC
Marc Schoenauer, INRIA-Saclay
Anne Auger, INRIA-Saclay

Lucas Bordeaux, MSRC
Michèle Sebag, CNRS

Parallel constraint programming and optimization for very large scientific data

- improve the usability of *Combinatorial Search* algorithms.
- automate the fine tuning of solver parameters.
- parallel solver: “disolver”



MoGo

CENTRE DE RECHERCHE COMMUN
INRIA
MICROSOFT RESEARCH

Adaptive Combinatorial Search for E-science

- **constraint programming**: learn instance-dependent variable ordering
- **evolutionary algorithms**: use multi-armed bandit algorithms and extreme values statistics
- **continuous search spaces:** use local curvature

Image and video mining for science and humanities

Jean Ponce, ENS
Andrew Blake, MSRC

Patrick Pérez, INRIA Rennes
Cordelia Schmid, INRIA Grenoble

Computer vision and Machine learning for:

- *sociology*: human activity modeling and recognition in video archives
- *archaeology and cultural heritage preservation*: 3D object modeling and recognition from historical paintings and photographs
- environmental *sciences*: change detection in dynamic satellite imagery

CENTRE DE RECHERCHE COMMUN
INRIA
MICROSOFT RESEARCH

Image and video mining for science and humanities

CENTRE DE RECHERCHE COMMUN
INRIA MICROSOFT RESEARCH

Sciences in track B

DDMF	computer algebra	hard sciences
Adapt. search	constraints, machine learning	hard sciences, biology
Reactivity	chi + visualisation	soft sciences, biology
I.V. mining	computer vision	humanities, environment

Future

CENTRE DE RECHERCHE COMMUN

INRIA MICROSOFT RESEARCH

Future

– 2nd anniversary: 28 January 2009 at Ecole Polytechnique

welcome   program   registration   detailed program   location

MICROSOFT RESEARCH
INRIA
JOINT
CENTRE

FORUM 2009

“ Recent research at the Joint Centre ”

28 JANUARY 2009

Open invitation - Registration Required - Please register before January 21st.

FORUM 2009

The Microsoft Research-INRIA Joint Centre has been created in October 2005. Three research projects started in May 2006, followed by four more in 2007 and 2008.

The research programme is divided into two tracks.

➡ A, **Software Security and Trustworthy Computing**, comprising 3 projects: Mathematical Components, Tools and Methodologies for Formal Specifications and for Proofs, and Secure Distributed Computations and their Proofs.

➡ B, **Computational Sciences and Scientific Information Interaction**, comprising 4 projects: Dynamic Dictionary of Mathematical Functions, ReActivity, Adaptive Combinatorial Search for E-Science, and Image and Video Mining for Science and Humanities.

At Forum 2009, all seven research projects will be presented with an update on progress and results. See program below.

Amphithéâtre Pierre Faurre
École Polytechnique
91128 Palaiseau

INFORMATION AND REGISTRATION

+33 1 69 35 69 70
forum2009@msr-inria.inria.fr
www.msr-inria.inria.fr/forum2009

INRIA
MICROSOFT RESEARCH

Future

- 30 resident researchers
- tight links with French academia (phD, post-doc)
- develop useful research for scientific community
- provide public tools (BSD-like license)
- become a new and attractive pole in CS research
- and source of spin off companies

CENTRE DE RECHERCHE
COMMUN

INRIA
MICROSOFT RESEARCH