ESHKOL v1.0-FOUNDATION — PRESS INFORMATION SHEET

For use by journalists and publications preparing coverage

BASIC INFORMATION

Product	Eshkol
Version	v1.0-foundation
Release Date	December 2025
License	MIT (open source)
Website	https://eshkol.ai
Source Code	https://github.com/tsotchke/eshkol
Company	Tsotchke Corporation
Creator	Tyrell Edward Umbach, CEO and sole engineer
Company Website	https://tsotchke.org

WHAT ESHKOL IS

Eshkol is a compiled programming language for mathematical computing. It is built on a Scheme (Lisp) foundation, compiles to native machine code via LLVM, and targets scientific computing, machine learning research, numerical simulation, and AI systems development.

KEY TECHNICAL FEATURES

Automatic Differentiation (Compiler-Native)

Eshkol is the first production programming language to integrate automatic differentiation at the compiler level rather than through external libraries.

Three differentiation modes:

- **Symbolic**: Compile-time AST transformation, zero runtime overhead
- **Forward-mode**: Dual number arithmetic for first derivatives
- **Reverse-mode**: Tape-based computational graphs for backpropagation

Capabilities:

- Nested gradients up to 32 levels deep
- Supports second-order optimization and meta-learning
- Zero overhead when differentiation is not used

Eight built-in vector calculus operators:

1. `gradient` — multivariate gradient (∇f)
2. `jacobian` — Jacobian matrix for vector functions
3. `hessian` — second derivative matrix
4. `divergence` — vector field divergence (∇·F)
5. `curl` — vector field curl (∇×F)
6. `laplacian` — scalar field Laplacian (∇²f)
7. `directional-derivative` — derivative in specified direction
8. `derivative` — single-variable first derivative

These are language primitives, not library functions.

Memory Management

System: Ownership-Aware Lexical Regions (OALR)

- Zero garbage collection
- Arena-based allocation with O(1) bump-pointer
- Deterministic deallocation at scope boundaries

- Compile-time ownership tracking prevents use-after-free
- Optional linear types for strict resource consumption
- 64KB block sizing optimized for CPU cache behavior

Result: Microsecond-scale worst-case execution time guarantees. No unpredictable pauses.

Native Compilation

- **Backend**: LLVM 10.0+
- **Targets**: x86-64, ARM64
- **Platforms**: macOS (Intel and Apple Silicon), Linux
- **Output**: Standalone native executables, object files for linking
- **Startup time**: Sub-50ms (compared to seconds for interpreted alternatives)

Homoiconicity Preserved in Compiled Code

Eshkol preserves Lisp's code-as-data property even after native compilation. Lambda expressions remain accessible as S-expression structures at runtime. This enables:

- Symbolic manipulation of compiled code
- Metaprogramming at native speed
- Runtime introspection of function definitions

No other compiled language maintains this property.

Type System

- **Foundation**: Homotopy Type Theory (HoTT)
- **Style**: Gradual typing — type errors produce warnings, not compilation failures
- **Features**:
  - Dependent types for dimensional correctness
  - Linear types for resource tracking
  - Polymorphic function types
  - Compile-time verification with zero runtime overhead when types are known

Language Foundation

- **Base**: R5RS/R7RS Scheme compatible
- **Special forms**: 39 (define, lambda, let/let*/letrec, if/cond/case/match, quote/quasiquote, pattern matching)
- **Built-in functions**: 300+
- **Macros**: Hygienic macros via syntax-rules with pattern matching
- **Tail call optimization**: Direct elimination and trampoline-based constant-stack recursion

IMPLEMENTATION DETAILS

Compiler written in	C17 (runtime) + C++20 (compiler)
Parser	Recursive descent, 5,487 lines
Type checker	Bidirectional Hindley-Milner with HoTT extensions, 1,561 lines
Codegen	19 modular LLVM code generation components
Test suite	200+ tests, all passing

Runtime representation:

- **Values**: 16-byte tagged structures for cache efficiency
- **Cons cells**: 32 bytes with complete type information
- **Closures**: 32 bytes with captured environments and preserved S-expressions

DEVELOPMENT TOOLS INCLUDED

- **eshkol-run**: Production compiler with executable and object file output
- **eshkol-repl**: Interactive REPL with LLVM JIT compilation

Standard library (300+ functions implemented in pure Eshkol):

- 60+ list operations

- 30+ string utilities
- JSON parsing and generation
- CSV processing
- Base64 encoding/decoding
- Linear algebra (LU decomposition, eigenvalue estimation, numerical integration)
- Functional combinators (compose, curry, partial application)

WHY ESHKOL WAS CREATED

Eshkol was created to address gaps in existing programming languages for AI and scientific computing:

1. **Python**: Dominant in machine learning but not designed for it. Garbage collection causes unpredictable latency. Automatic differentiation exists only through external frameworks.
2. **Julia**: Partially homoiconic but lacks compiler-level automatic differentiation and deterministic memory management.
3. **Scheme/Lisp dialects**: No existing high-performance Scheme suitable for scientific computing. None have advanced type systems or native differentiation.
4. **Rust**: Deterministic memory but no automatic differentiation, not designed for mathematical computing.

Eshkol was designed from first principles for mathematical and scientific computing in the era of machine learning and AI.

INTERNAL USE AT TSOTCHKE CORPORATION

Eshkol serves as foundational infrastructure for Tsotchke Corporation's products and research:

- **Selene (qLLM)**: A semiclassical quantum large language model using geometric principles. Achieves 10-1000× parameter efficiency over conventional transformers through mixed-curvature embeddings (hyperbolic × spherical × Euclidean manifolds). Eshkol's native differentiation and geometric computing capabilities enable this architecture.
- **Moonlab**: A quantum computing simulator with Bell-verified quantum behavior, exceeding 10,000× performance on Apple Silicon. Eshkol is used for circuit transpilation and optimization.

- **Quantum computing research**: Topological circuit transpilation and other advanced quantum computing applications.

The language was developed to solve real problems encountered in building these systems, not as a theoretical exercise.

TARGET USERS

- Machine learning researchers
- Scientific computing engineers
- Numerical simulation developers
- Robotics and real-time systems engineers
- Embedded AI/ML systems developers
- University research groups
- Anyone requiring deterministic performance with mathematical computing capabilities

QUOTE FROM CREATOR

> "We built Eshkol because the future of computing requires a language that treats mathematical concepts—automatic differentiation, geometric operations, deterministic performance—as fundamental rather than afterthoughts. We're releasing it because researchers and engineers building the next generation of intelligent systems deserve tools designed for this era, not adapted from the last one."
>
> — **Tyrell Edward Umbach**, CEO of Tsotchke Corporation

ABOUT TSOTCHKE CORPORATION

Tsotchke Corporation develops infrastructure for quantum computing, geometric artificial intelligence, and scientific computing applications.

Key technologies:

- Eshkol programming language
- Selene semiclassical quantum LLM
- Moonlab quantum simulator
- Project Neo-Millennium quantum hardware initiatives

Mission: Advance computing through mathematical rigor and novel architectures.

Contact: team@tsotchke.org

TECHNICAL COMPARISONS

Feature	Eshkol	Python/JAX	Julia	Rust	Racket
Native AD	✓ (3 modes, compiler-level)	Library	Library	✗	✗
Zero GC	✓	✗	✗	✓	✗
Homoiconicity	✓ (native compiled)	✗	Partial	✗	✓ (interpreted)
Native compilation	✓ (LLVM)	✓ (XLA)	✓	✓	✗
Dependent types	✓ (HoTT)	✗	✗	✗	✗
Vector calculus ops	✓ (8 built-in)	Library	Library	✗	✗
Deterministic latency	✓	✗	✗	✓	✗

RESOURCES FOR JOURNALISTS

Resource	Location
Technical documentation	https://eshkol.ai
Source code	https://github.com/tsotchke/eshkol

Resource	Location
Language reference	`ESHKOL_V1_LANGUAGE_REFERENCE.md` in repository
Architecture guide	`docs/ESHKOL_V1_ARCHITECTURE.md`
Media inquiries	team@tsotchke.org

This document may be used freely for preparing news coverage of Eshkol v1.0-foundation.