Optimizing Strategies for Telescoping Languages: Procedure Strength Reduction and Procedure Vectorization

presentation by

Arun Chauhan

joint work with

Ken Kennedy

Motivation

- Shortage of programmers
  - increasing application demands
  - rapidly changing architectures
  - need programmers for scientific applications too

Motivation

- Shortage of programmers
  - increasing application demands
  - rapidly changing architectures
  - need programmers for scientific applications too
- High Performance programming is hard
  - increasingly a specialized activity
  - more complex architectures
  - more high performance applications

One Solution

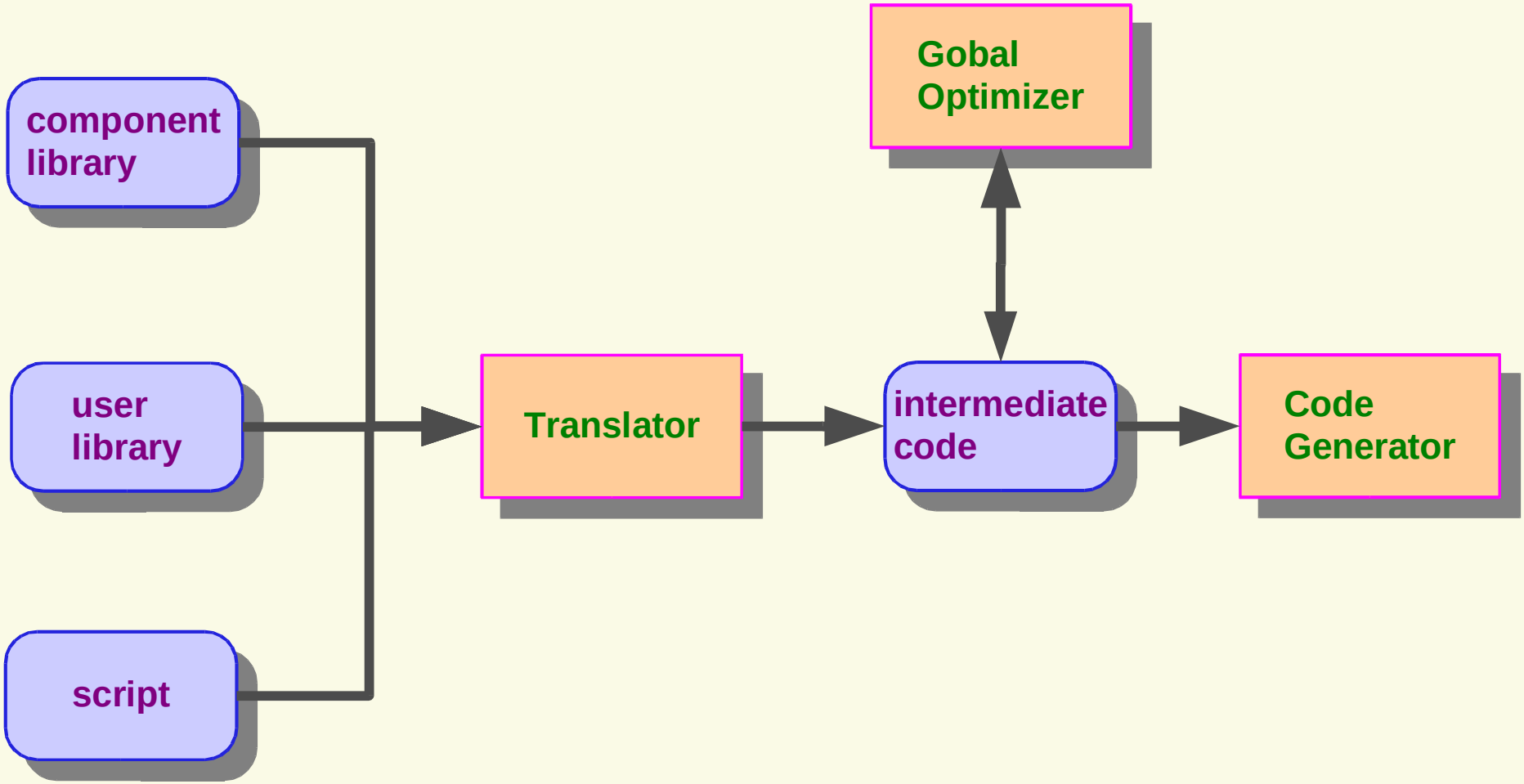
- Make end−users application developers
  - language should be high level
  - should provide domain−specific features
  - must have effective and efficient compilers

One Solution

- Make end−users application developers
  - language should be high level
  - should provide domain−specific features
  - must have effective and efficient compilers
- Scripting systems like MATLAB exist
  - very popular with end−users
  - lack effective and efficient compilers

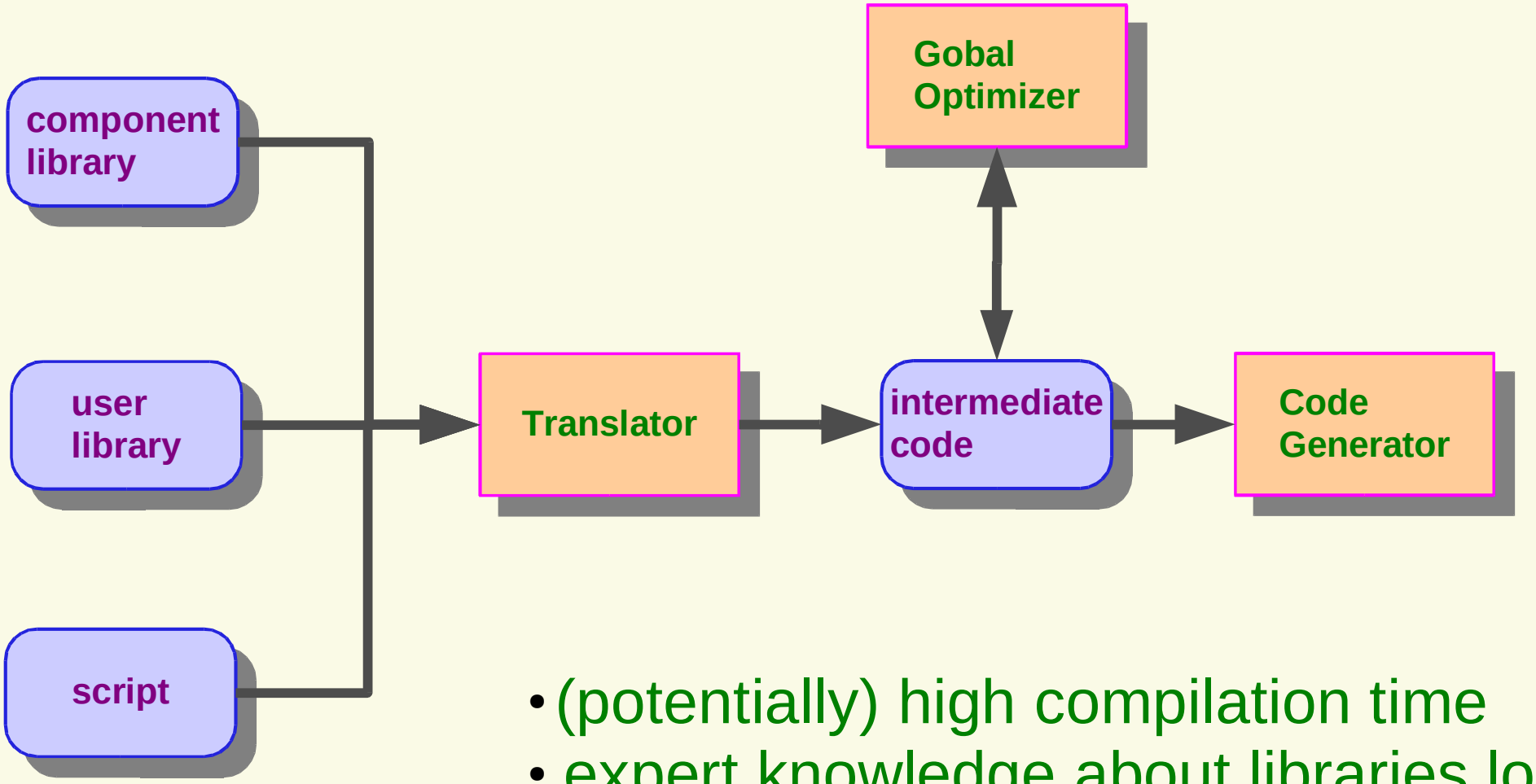
Existing Approaches:

based on transforming to lower level languages



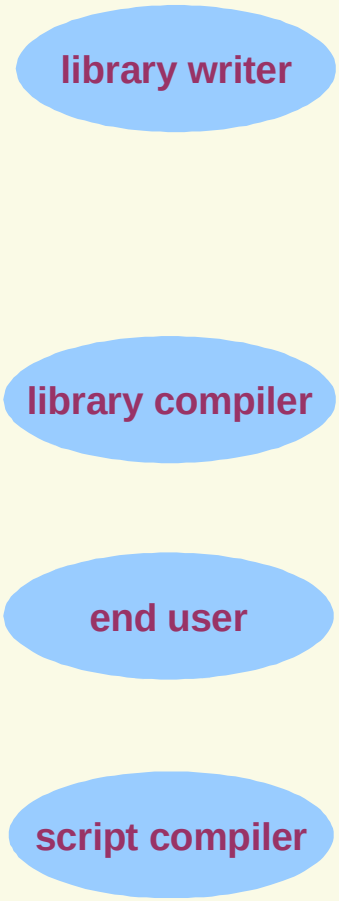
Existing Approaches:

based on transforming to lower level languages

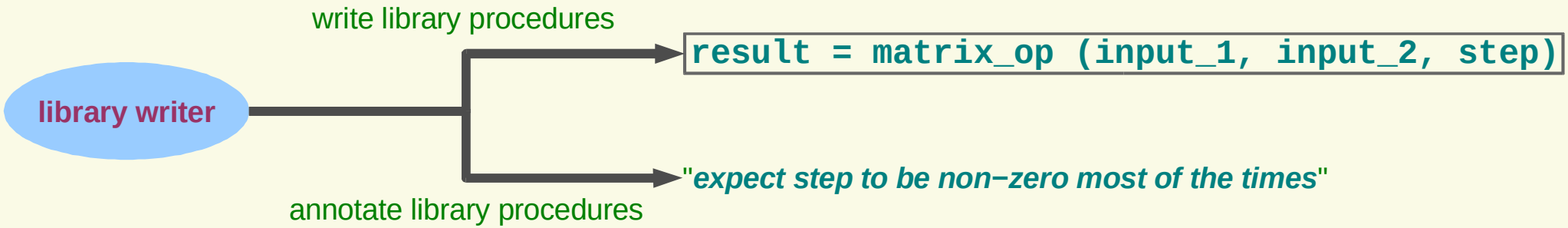


- (potentially) high compilation time
- expert knowledge about libraries lost

Telescoping Languages Approach



Telescoping Languages Approach

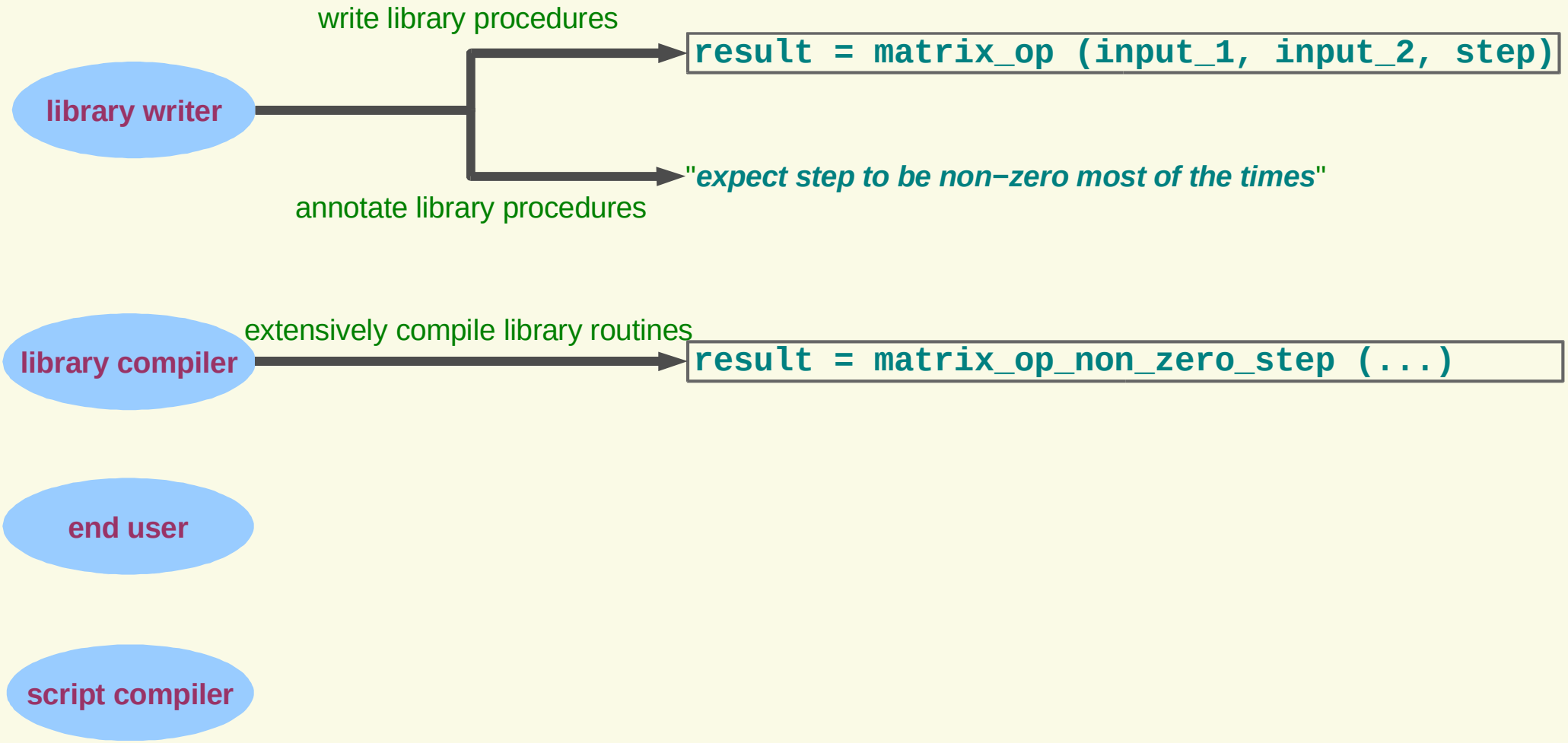


library compiler

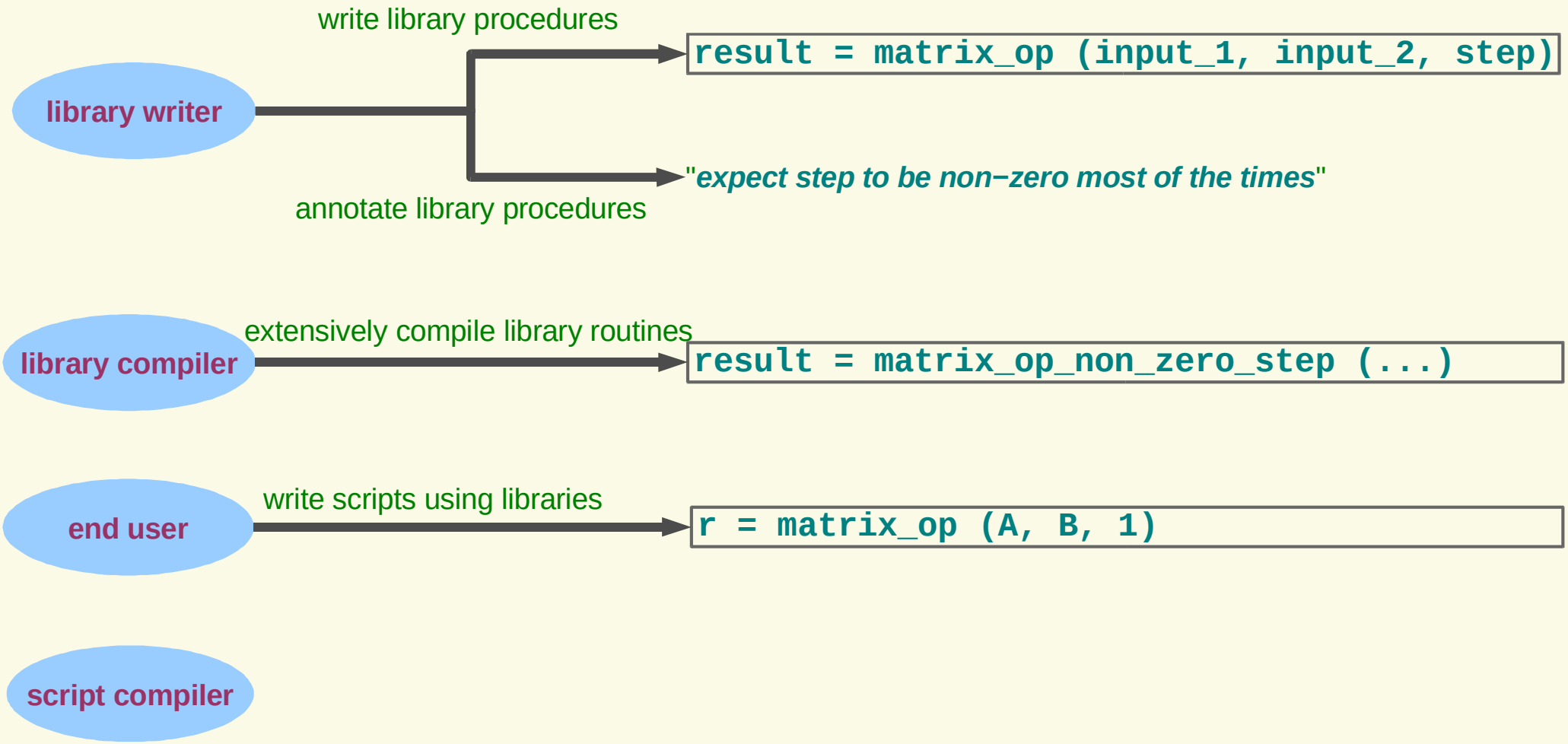
end user

script compiler

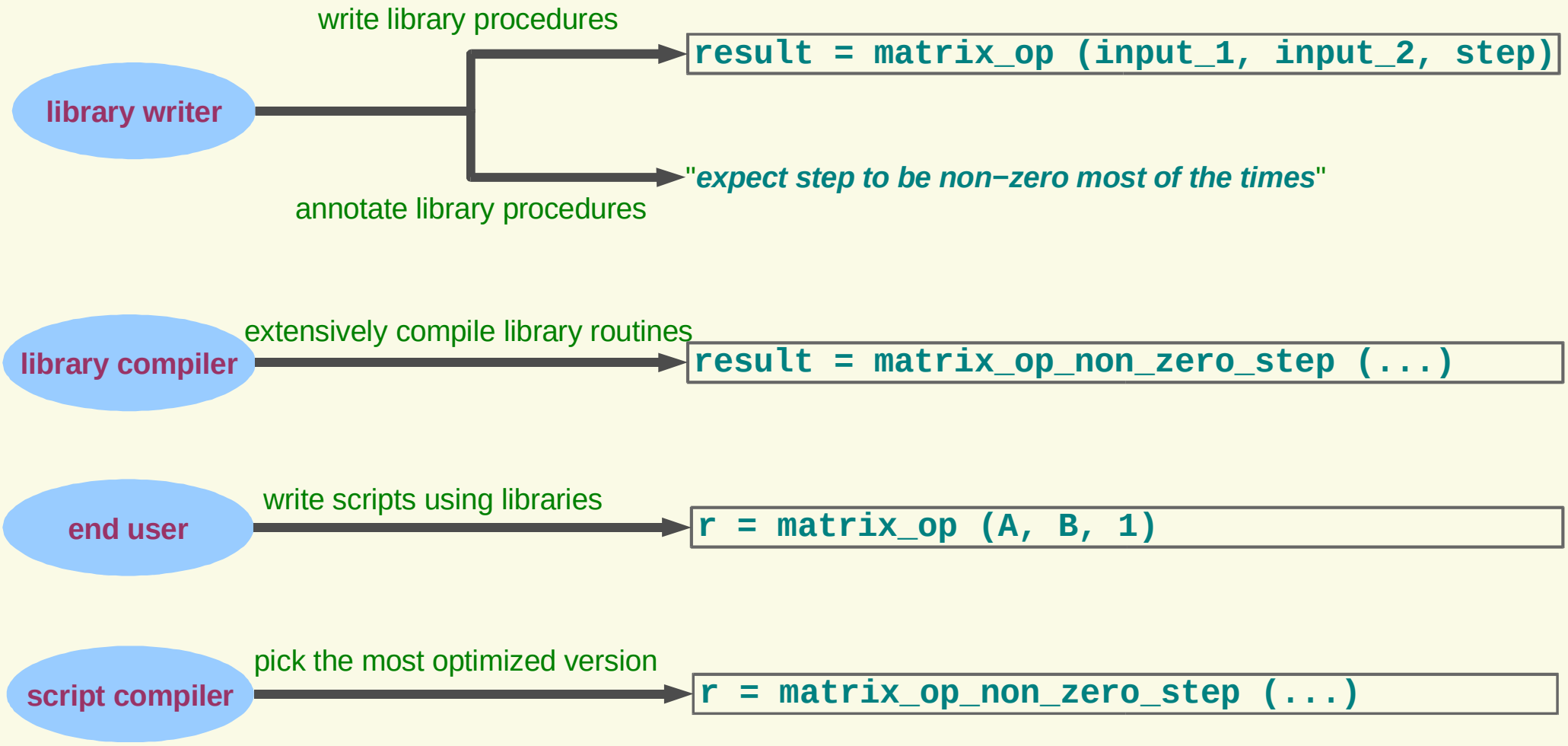
Telescoping Languages Approach



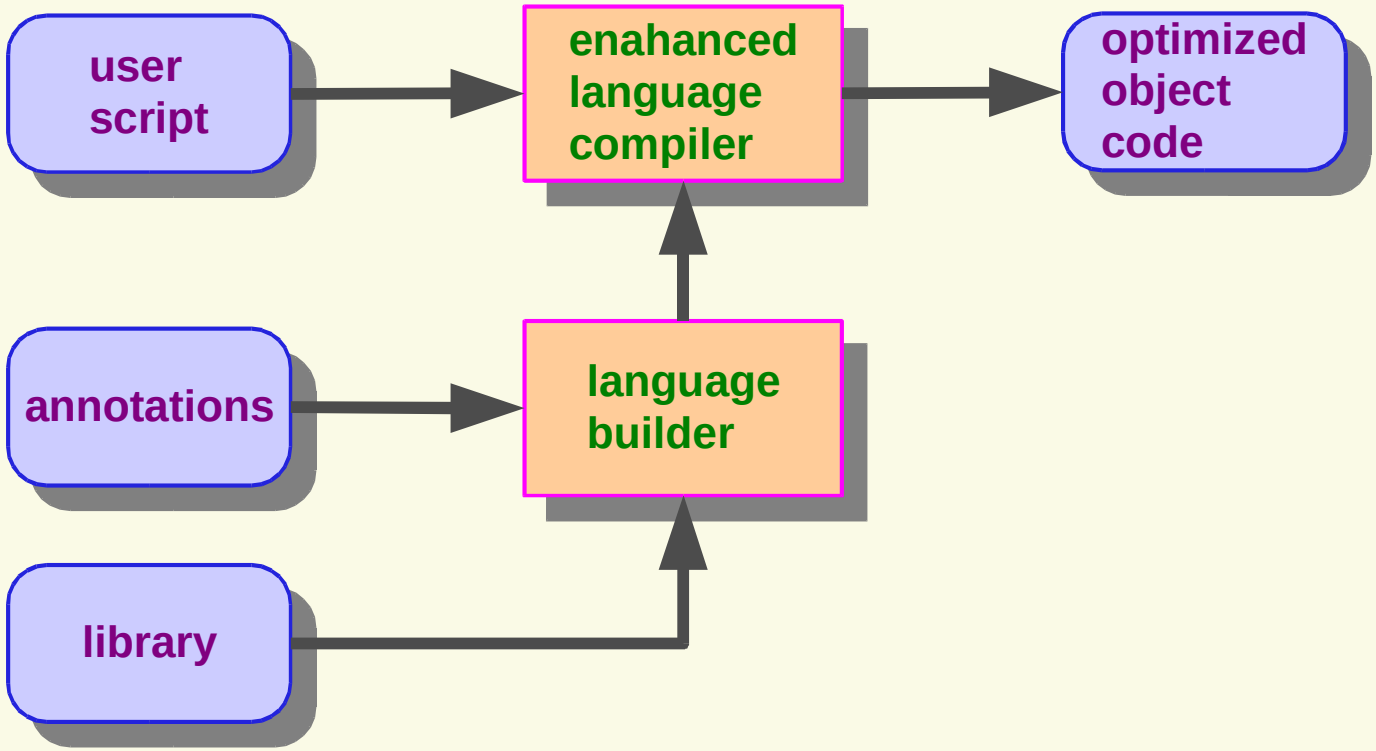
Telescoping Languages Approach



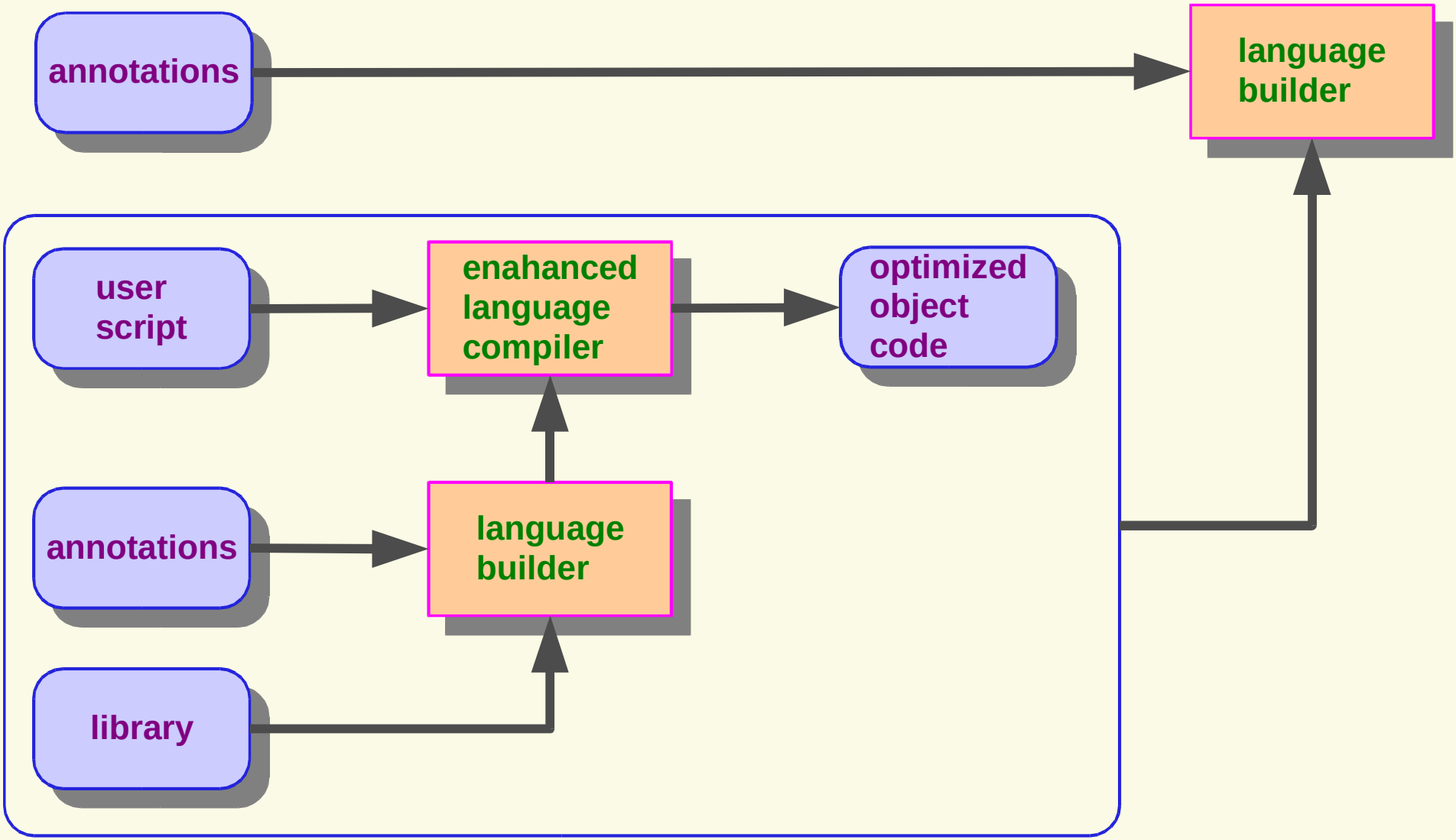
Telescoping Languages Approach



Telescoping Languages Approach



Telescoping Languages Approach



Application to DSP

- Digital Signal Processing is an important application area
- Matlab extremely popular with DSP researchers
- Researchers frequently rewrite their code in C / C++
  - to fit it onto embedded systems
  - to be accepted in their community

Study of DSP Applications

- Real DSP applications used in the ECE dept.
- Long running (several hours)
- Transformed by hand
- Run under Matlab environment

Relevant Optimizations

- Vectorization
- Common sub−expression elimination
- Array pre−allocation
  - using the **`zeros`** call

Relevant Optimizations

- Vectorization
- Common sub−expression elimination
- Array pre−allocation
  - using the **`zeros`** call

Novel Optimizations

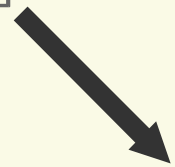
- Procedure Strength Reduction
- Procedure Vectorization

Strength Reduction

```
for i = n0:n0+N
  ....
  x = i * c;
  y = g(x);
  ....
end
```

Strength Reduction

```
for i = n0:n0+N
  ....
  x = i * c;
  y = g(x);
  ....
end
```



```
x = n0 * c;
for i = n0:n0+N
  ....
  y = g(x);
  x = x + c;
  ....
end
```

Procedure Strength Reduction

- Motivation
  - procedure calls inside loop
  - several arguments typically invariant
- Key
  - move invariant computations into init part
  - do incremental computations inside loop

Procedure Strength Reduction

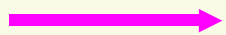
- Motivation
  - procedure calls inside loop
  - several arguments typically invariant
- Key
  - move invariant computations into init part
  - do incremental computations inside loop

```
for i = 1:N
    f (c1, c2, i, c3)
end
```

Procedure Strength Reduction

- Motivation
  - procedure calls inside loop
  - several arguments typically invariant
- Key
  - move invariant computations into init part
  - do incremental computations inside loop

```
for i = 1:N
    f (c1, c2, i, c3)
end
```



```
f_init (c1, c2, c3)
for i = 1:N
    f_iter (i)
end
```

Procedure Vectorization

- Motivation
  - vectorized operations more "optimizable"
  - expensive library call boundaries
- Key
  - interchange loops and procedure calls

Procedure Vectorization

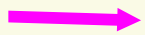
- Motivation
  - vectorized operations more "optimizable"
  - expensive library call boundaries
- Key
  - interchange loops and procedure calls

```
for i = 1:N
    f (c1, c2, i, A[i])
end
 ....
function f (a1, a2, a3, a4)
   <body of f>
```

Procedure Vectorization

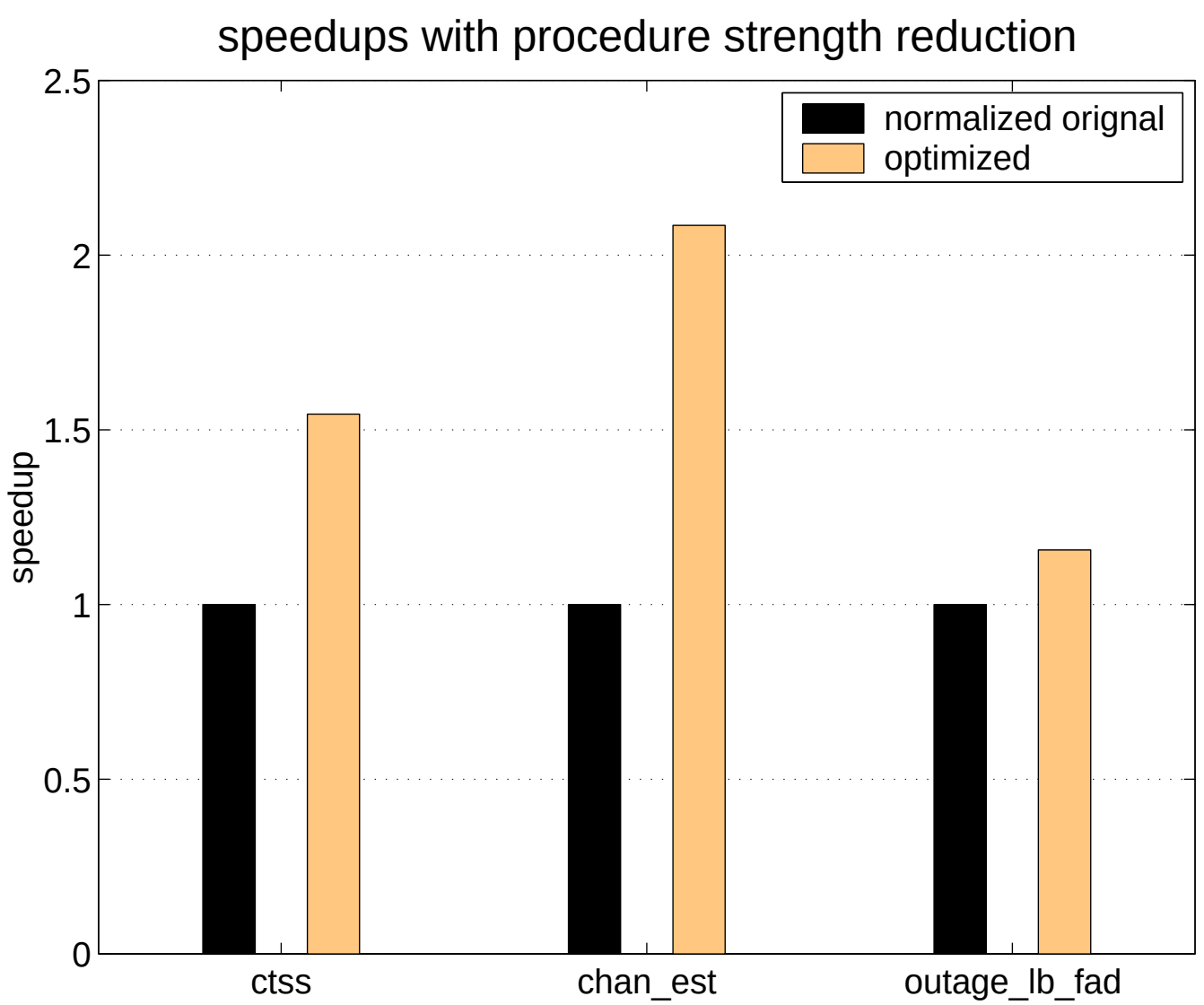
- Motivation
  - vectorized operations more "optimizable"
  - expensive library call boundaries
- Key
  - interchange loops and procedure calls

```
for i = 1:N
    f (c1, c2, i, A[i])
end
 ....
function f (a1, a2, a3, a4)
   <body of f>
```

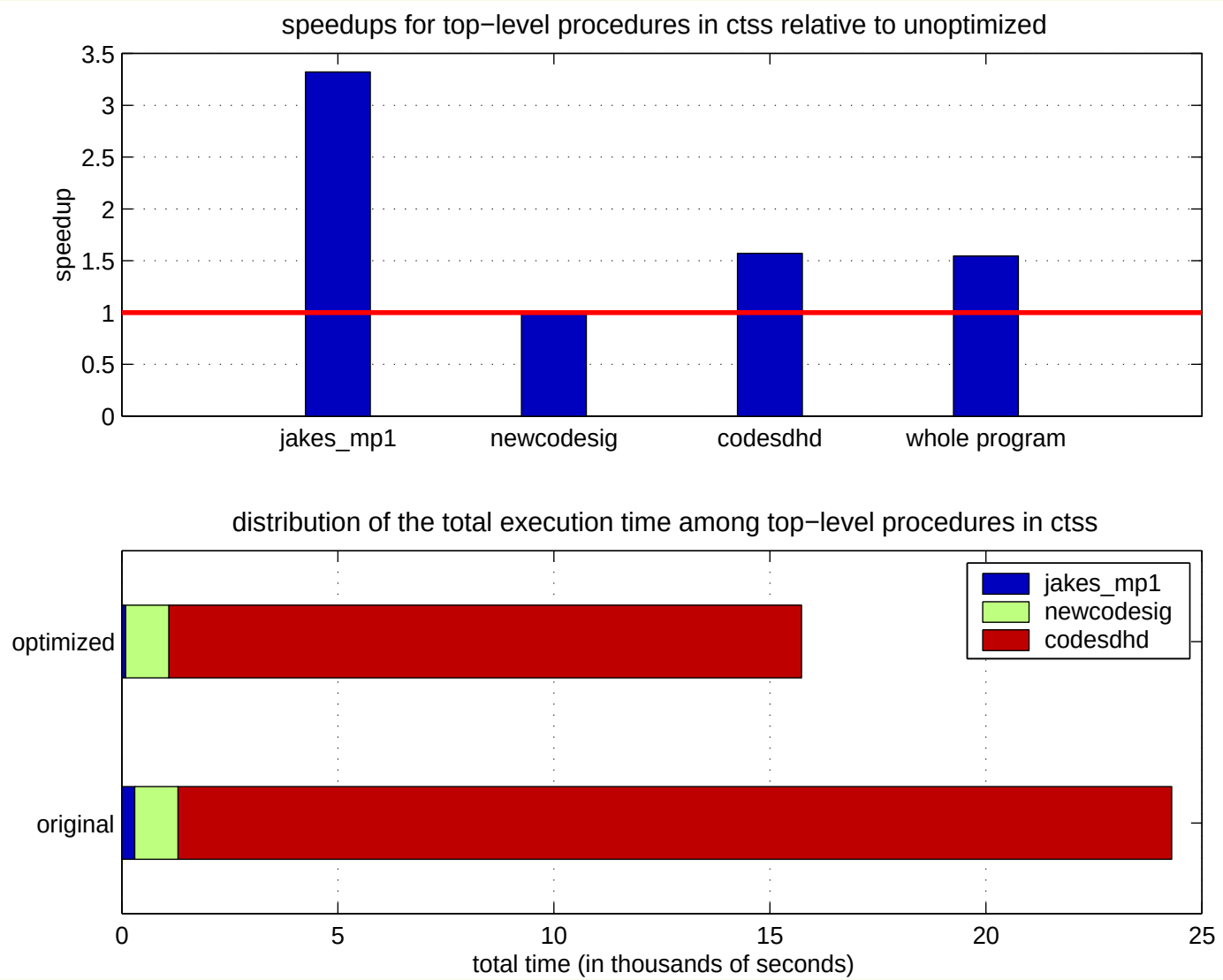


```
f_vect (c1, c2, [i:N], A[1:N])
 ....
function f_vect (a1, a2, a3, a4)
  for i = 1:N
     <body of f>
  end
```

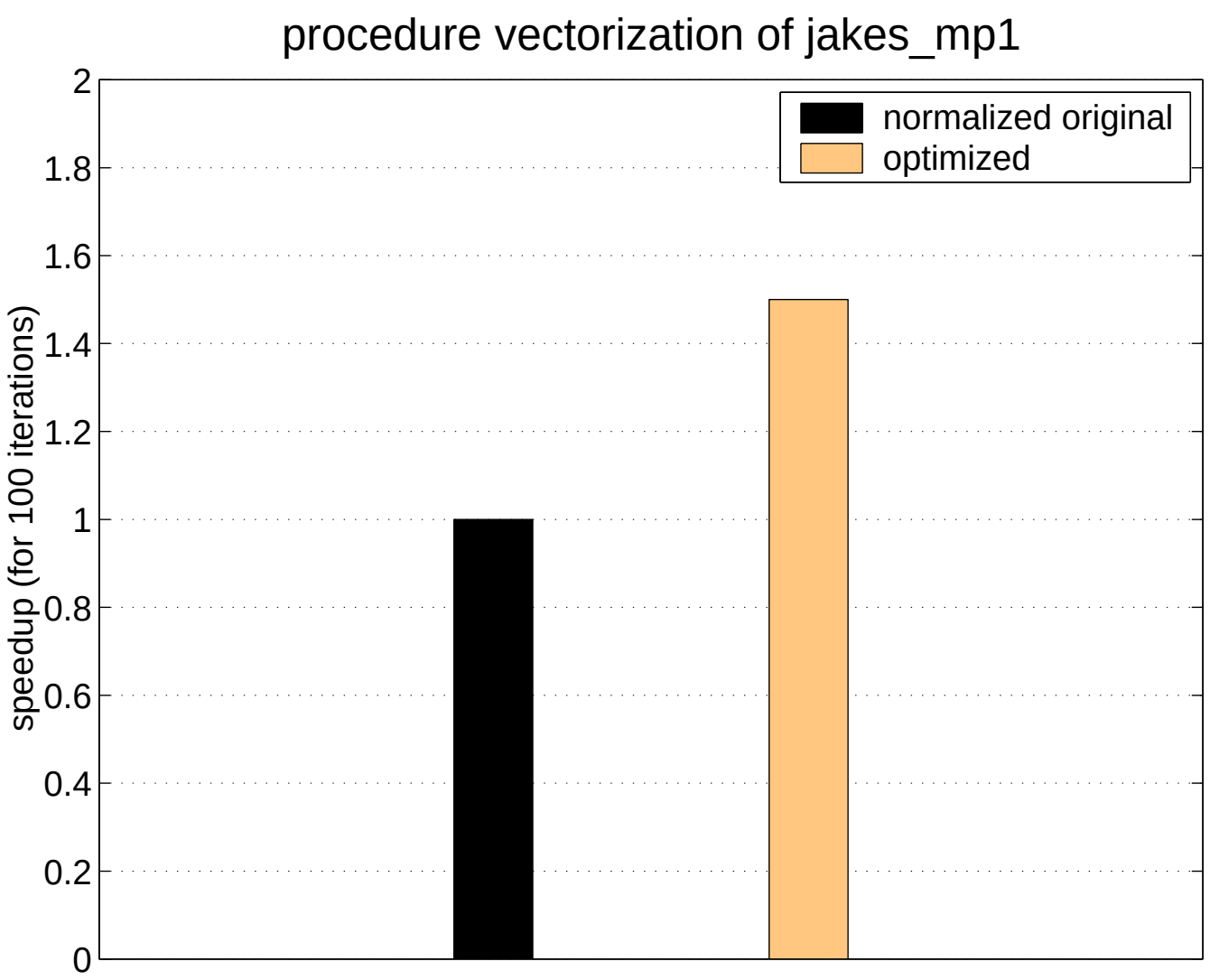
Evaluation Results: Procedure Strength Reduction



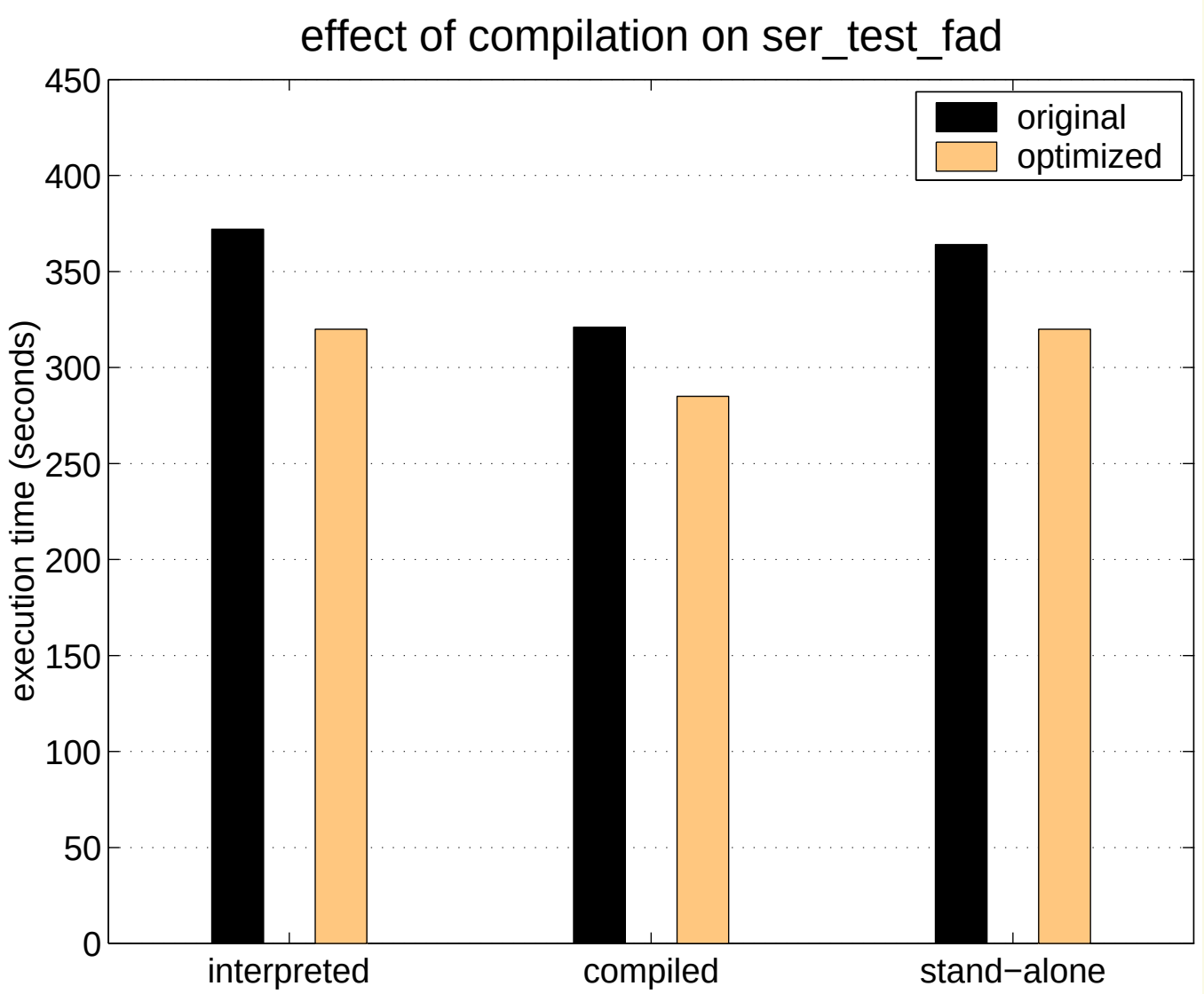
ctss: Procedure Strength Reduction



ctss: Procedure Vectorization



outage_lb_fad: Effect of Compilation



Conclusion

- Telescoping Languages approach
  - libraries optimized as primitive operations
  - fast compilation of user scripts
- Application to DSP programs
  - identified relevant optimizations
    - vectorization, common sub−expression elimination, pre−allocation
  - two novel optimizations
    - procedure strength reduction and procedure vectorization
    - 10% – 50% application level gain