APPROXIMATIONS FOR MONEY-MARKET COMPOUNDING
A SAMPLE HOMEWORK

BRIAN K. BOONSTRA

1. Introduction

A dollar invested at a given rate r is expected to increase according to a daily compounded rate r, which we can think of as the risk-free rate available to the investor, e.g. 3-month LIBOR. We investigate various approximations to true returns to the investor. These approaches are

- The continuous compounding formula, resulting from an integral
- Various Riemann sums of the logarithm, corresponding to different approaches to approximating the same interval
- The same approaches, with "boundaries" set at times when the slope of r changes from one level to the next.

2. Functions Used

2.1. **Required functions.**

The assignment specifications require the following 8 functions:

- `compInt(startingVal, startTime, endTime, riskfree)` for using the exact continuously compounded integral. This function returns a single variable: the integral value.
- `compIntBdry(startingVal, startTime, endTime, riskfree)` as above, with separate computation at the boundaries in the domain of `riskfree`. This function returns a single variable: the integral value.
- `leftRiemann(startingVal, startTime, endTime, riskfree, M)` for using the Riemann sum based on left-hand endpoints. This function returns a single variable: the integral value.
- `leftRiemannBdry(startingVal, startTime, endTime, riskfree, M)` as above, with separate computation at the boundaries in the domain of `riskfree`. This function returns a single variable: the integral value.
- `rightRiemann(startingVal, startTime, endTime, riskfree, M)` for using the Riemann sum based on right-hand endpoints. This function returns a single variable: the integral value.
- `rightRiemannBdry(startingVal, startTime, endTime, riskfree, M)` as above, with separate computation at the boundaries in the domain of `riskfree`. This function returns a single variable: the integral value.
- `midRiemann(startingVal, startTime, endTime, riskfree, M)` for using the Riemann sum based on midpoints. This function returns a single variable: the integral value.
- `midRiemannBdry(startingVal, startTime, endTime, riskfree, M)` as above, with separate computation at the boundaries in the domain of `riskfree`. This function returns a single variable: the integral value.

where startingVal is a number representing the starting dollar amount in the account, M is the minimum step count, startTime and endTime (specified in days) define the time over which the account is held, and riskfree is a list of two-element tuples with the daily risk-free rates r in to be linearly interpolated in the first tuple elements and the times (specified in days) in second elements of tuple.

2.2. **Helper functions.** Rather than copy-and-paste too much code, I wrote a single function

```
anyRiemann(startingVal, startTime, endTime, riskfree, M, basedOn, useBoundaries)
```

which uses the left-, right-, or mid-point Riemann scheme according to whether the basedOn variable is -1, 0, or 1, and ignores or uses boundaries in the riskfree rate domain according to whether useBoundaries is True or False. The "required functions" are then written as nearly empty dummy functions, that access this main function as appropriate. The outputs of anyRiemann are a dictionary:

- key 'val': the value returned by the requested Riemann sum
- key 'exactVal': the exact value, startingVal $\prod_{t=t_1}^{t_N}(1+r_t)$
- key 'err': the error in the Riemann sum value

Also, to aid the analysis of error, I created a function that reports on the rate of convergence and makes a log-log plot of error versus M

```
[convergenceRate, errs, compIntErr] = anyRiemannErrorAnalysis(startingVal, startTime, ...
    endTime, riskfree, M, basedOn, useBoundaries)
```

Here, the input M is expected to be a two or more element vector of minimum step counts to use The outputs are a graph, and the following dictionary:

- key 'convergenceSlope': the polynomial "rate" of convergence, inferred from the errors at the given step counts
- key 'errs': the errors in the Riemann sum value at the given step counts
- key 'compIntErr': the error in the continuously compounding calculation

3. Test Suite

Checking my functions against the provided test suite demonstrates that they match the required API, and that they agree with most of the provided test cases. However, the test case test12 failed for me. I think that test has a bug because it should be testing a negative interest rate case but appears to be using absolute value of the given interest rates.

4. Range of Inputs

I tested the speed, accuracy and efficiency of these approximation approaches for the following problems:

(1) Annualized rates of 4%, i.e. `riskfree` $= [(0.04/365, 0), (0.04/365, 100000)]$. This simple low-interest-rate case should have relatively small errors.
(2) Annualized rates of 8% at 600 days, then 15% at that plus 900 days, i.e. `riskfree` $= [(0.08/365, 0), (0.15/365, 600), (0.15/365, 900)]$. This should show somewhat higher errors because of the relatively high interest rates and long duration.
(3) Annualized rates of 10% to start, then 5% after 500 days, and 15% after 500 more days and 10% at 1800 days, i.e. `riskfree` $= [(0.1/365, 0), (0.05/365, 500), (0.15/365, 1000), (0.1/365, 1500)]$. This is likely the most error-prone case of all because of the wildly varying rate, it's high average level, the long duration, and the presence of the three distinct interest rate domains.

I tested minimum stepcounts of

```
M = [3, 8, 12, 15,20, 30,40,50, 60]
```

and

```
M =[3, 8, 15,30,40,50, 60, 70, 80, 100, 150, 200, 400, 1000, 2000, 2500, 5000, 10000]
```

In each case, I started the account on day 100 and ended it on day 1400.

Students: In general your own analyses should cover a wider variety of inputs than this!

5. Analysis

Ordinary Riemann sums work as pictured, so we clearly expect better performance from the midpoint version of our integral. In addition, since in our linearly interpolated case

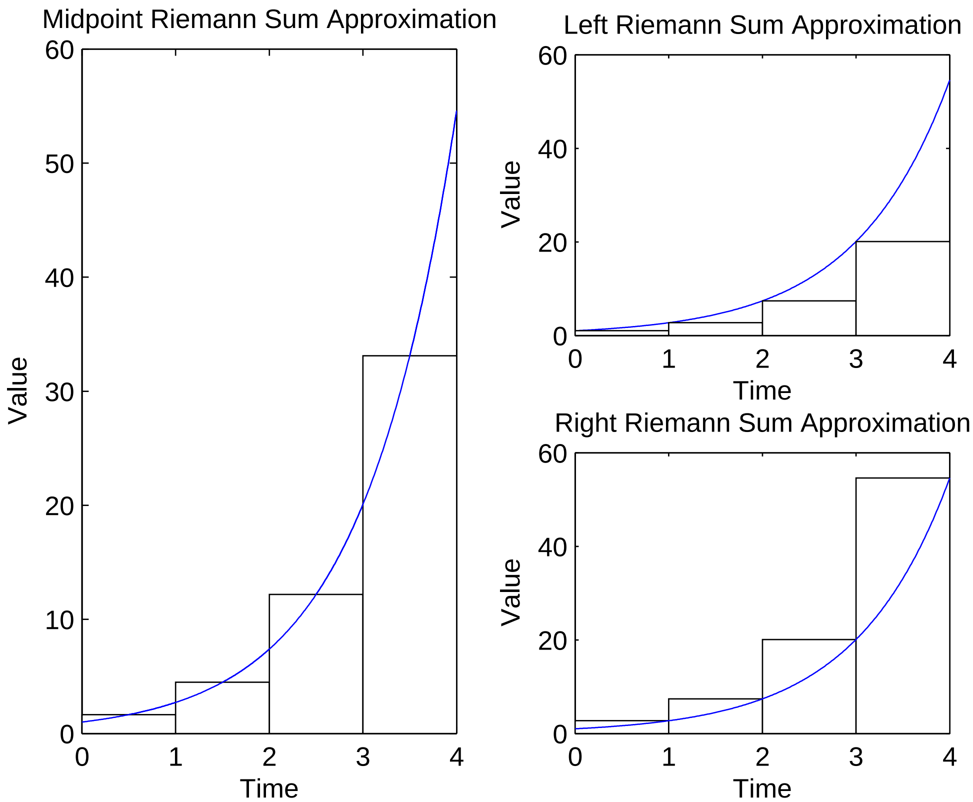
$$\int_{T-\Delta t}^{T} \frac{dV}{V} = \int_{T-\Delta t}^{T} r(t)dt = \int_{T-\Delta t}^{T} r(T-\Delta t) + (r(T) - r(T-\Delta t))\frac{t-(T-\Delta t)}{\Delta t}dt$$

we have

$$\log\left(\frac{V(T)}{V(T-\Delta t)}\right) = \frac{r(T) + r(T-\Delta t)}{2}\Delta t$$

so we expect our midpoint formula to coincide with the continuously compounding formula whenever the integration boundary points coincide.

In every case, I found that when step counts get too high, accuracy begins to decrease. This is seen in the plots, where absolute values of errors start increasing when M gets large, and is also seen in the fact that the inferred rate of convergence gets smaller when step counts are allowed to get too big. These larger errors are expected, since high step counts begin to approximate continuous compounding more closely than the true daily compounding.



I found "continuous compounding" to give me errors of 22% and more, which leads me to suspect a bug in my code. I tried using extremely long and extremely short tenors, and found the error was correlated with tenor, so it would seem the formula is wrong.

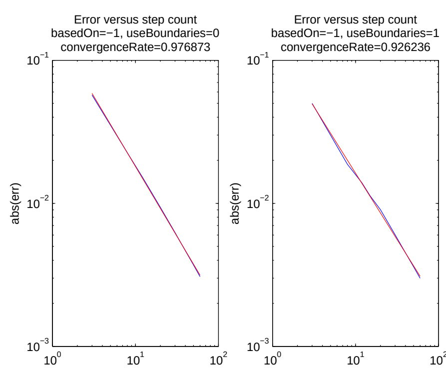


FIGURE 1. Left endpoints, reasonable M

5.1. **Left-endpoint integration.** We see that using the left endpoint works reasonably well, so long as M is not too large.

When M is too large, errors are introduced.

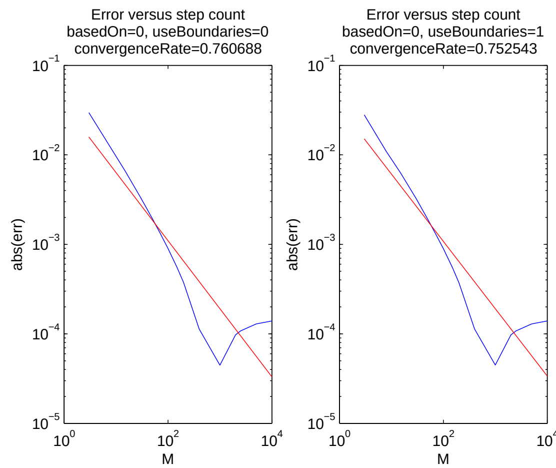


FIGURE 2. Left endpoints, unreasonable M

5.2. **Right-endpoint integration.** We see similar behavior for too-large M using right endpoints, but the convergence appears much quicker otherwise.

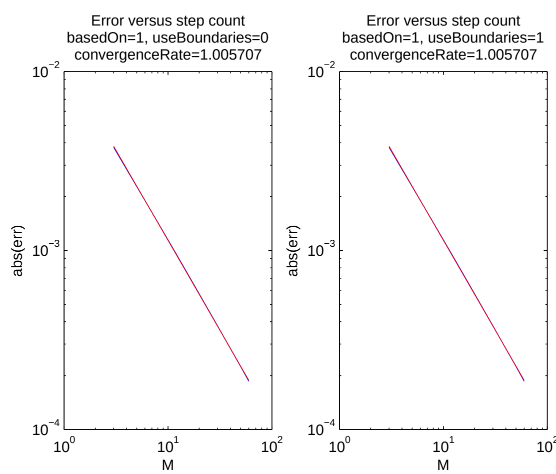


FIGURE 3. Right endpoints, reasonable M, easy interest rates

5.3. **Midpoint integration.** We see similar behavior for too-large M using the average, and the convergence is similar to the left endpoint.

Students: Some plots and narratives have been left out here for brevity

5.4. **Special treatment of boundary points.** As we can see in the above plots, inclusion of the boundary points (left sides versus right sides) makes little difference. If rates were considered piecewise constant rather than piecewise linear, this might be more important.

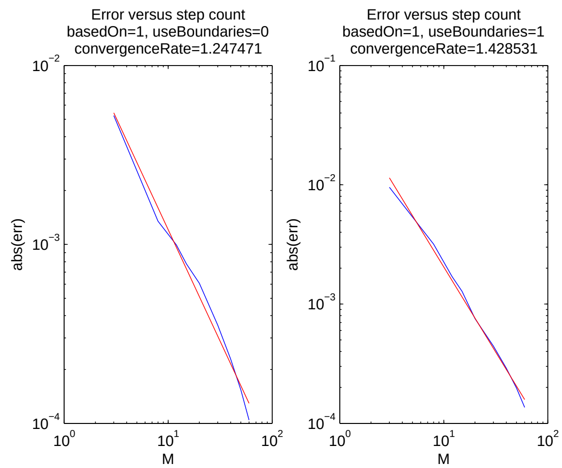


FIGURE 4. Right endpoints, reasonable M

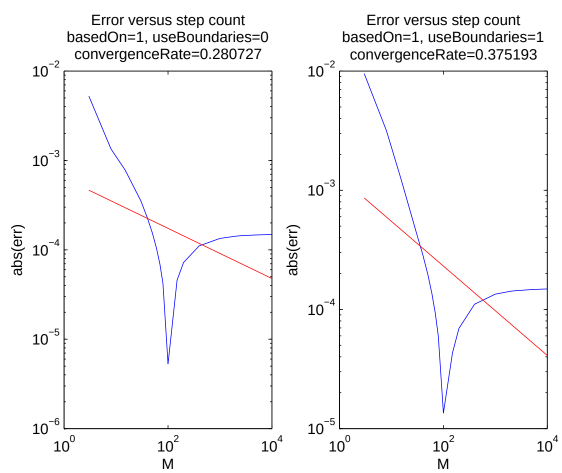


FIGURE 5. Right endpoints, unreasonable M

6. Comparison of results

From the plots we can see that as expected, the more difficult interest rate curves are tougher to match. Clearly the right endpoints do the best.

7. Answers to Questions Posed

This homework assignment posed two questions:

7.1. **Is the extra accuracy achieved from midpoint integration worthwhile?** No, I did better with the right endpoint. Presumably this is because of the exponential nature of the problem, so controlling errors on the bigger, right, side of the exponential function is more important.

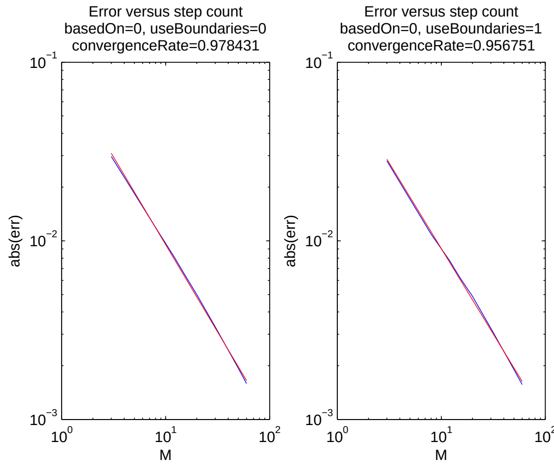


FIGURE 6. Right endpoints, reasonable M

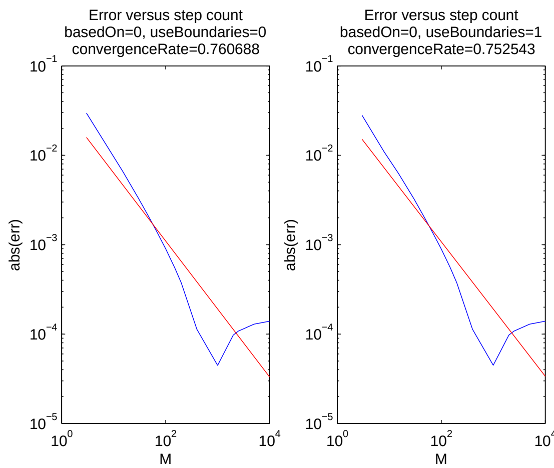


FIGURE 7. Right endpoints, unreasonable M

7.2. **Does the convergence appear polynomial?** Yes, I observed approximately M^{-1} rate of convergence for every case, at least in the applicable region.

7.3. **What other approximations might you suggest?** Perhaps continuous compounding could be given some sort of convexity correction, to make it more accurate.

8. Further Investigation

Though this analysis has provided an investigation of the given approximation techniques, it leaves certain questions unanswered. What special treatments should be given to holidays and weekends?

The approximations given allow for time boundaries to fall on fractional days. Perhaps it would be more sensible to force all boundaries to fall in integral days. This would eliminate the problem of too many timesteps creating inaccuracy.

9. Conclusion

No approximation technique investigated here is particularly good, particularly because of the danger of using too many timesteps, though I have suggested a fix for that particular problem. All converged reasonably quickly, but the computations saved are not likely to be all that significant for reasonable tenors. If performance is important, a matrix of discount factors could be maintained in order to compute cash rollups in a lookup table.